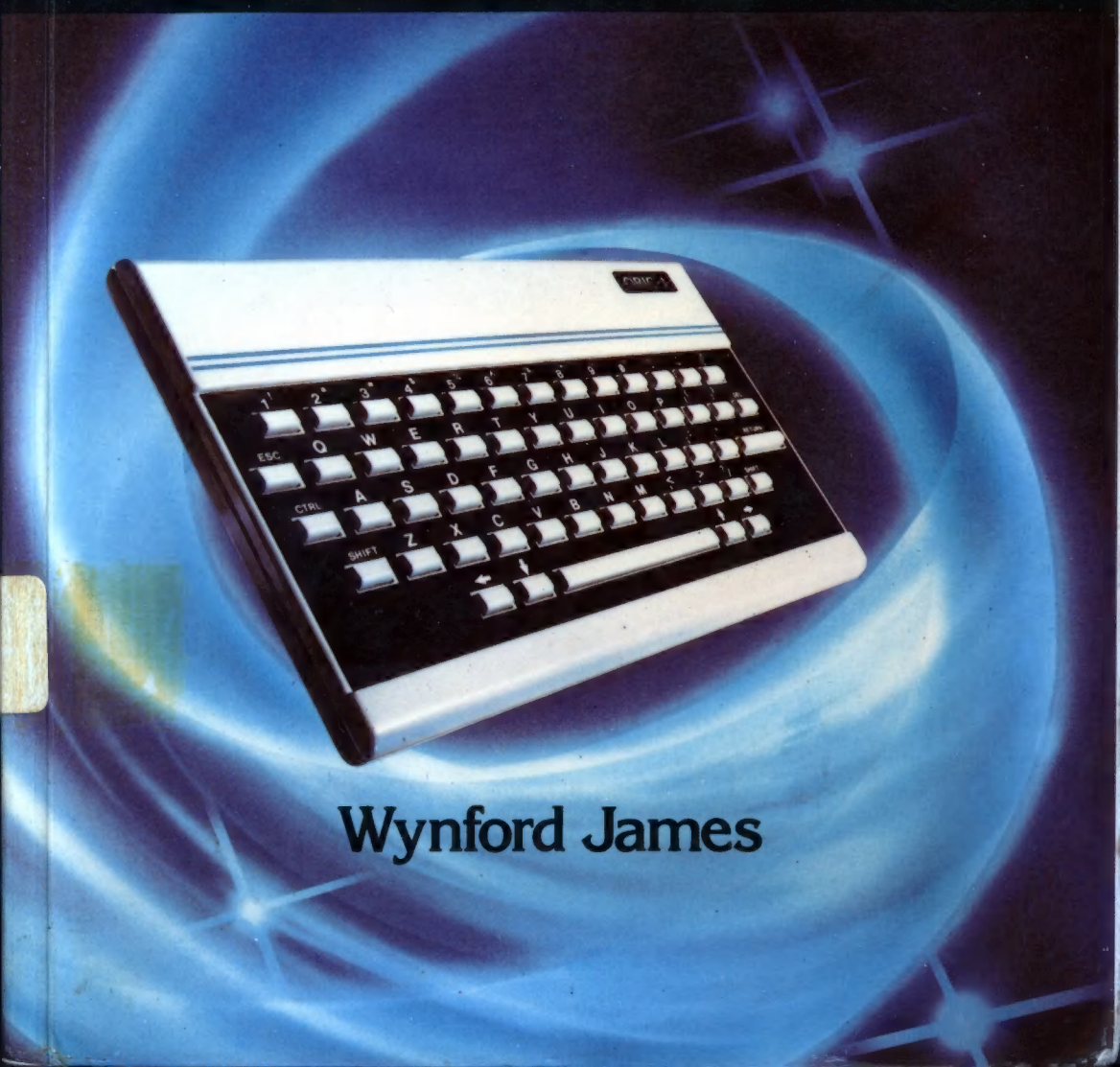
20 GAMES FOR THE ORIC-1



Wynford James

20 Games for the Oric-1

20 Games for the Oric-1

Wynford James

MICRO PRESS

First published 1983 by
Micro Press
Castle House, 27 London Road
Tunbridge Wells, Kent

British Library Cataloguing in Publication Data

James, Wynford
20 games for the Oric-1.
1. Oric-1 (Computer)—Programming
I. Title
001.64 QA76.8.0/

ISBN 0-7447-0003-5

Typeset by Keyset Composition, Colchester, Essex
Printed by Mackays of Chatham Ltd

Contents

Introduction

The Oric is a fine machine, with excellent sound and colour facilities. Unfortunately the manual supplied with the Oric is rather uninformative about some of its capabilities, so we will begin by explaining some of the statements which are frequently used in graphics programs.

What you see on your television screen when you are typing in or running a program is actually part of the computer memory. Just as any program is stored in the computer memory, so data can be stored in the part of the memory which is visible on-screen. To store data there we need one vital piece of information — what are the memory locations used for the screen display? Consulting the manual reveals that in text or lores mode, the screen occupies the locations from #BB8O to #BFEO. The # at the start of these numbers means that they are in hexadecimal. If you don't know what that means, don't worry, but it is important that you do not miss the # from the start of the numbers whenever they are used in a program.

Now that we know the screen memory locations we can have a look at the method for getting things to appear on-screen.

POKE is normally described as a command which changes the contents of a memory location. If you type POKE #BC84,65 the number 65 will be placed at memory location #BC84, which just happens to be one of the screen memory locations. Will we see the number 65 appearing in the middle of the screen? No, because to the Oric, each number is also what is called a character code, and the number 65 is the character code for A. By typing POKE #BC84,65 we cause the letter A to appear in the middle of the screen. A list of some of the Oric character codes is given in Appendix D of the manual.

So we get characters on-screen using POKE. But how do we know when we have shot a space invader, scored a hit on a sub, or bumped into a wall? For this we use PEEK.

PEEK enables us to examine the contents of a memory location

— in other words, find out what is at a particular address. When running games programs, we often need to PEEK to a memory location to see what character code is stored there. Returning to our earlier example, if we typed POKE #BC84,65: PRINT PEEK #BC84 the Oric will print 65, because it is telling us what is stored at the memory location where we just placed the number 65.

Suppose we were playing a simple game where we fired at space invaders which were dropping down the screen towards us. Every time we fired a 'bullet' at a space invader we would have a chance of hitting it. Before we moved the bullet to this new position, we would PEEK that memory location to see what is there. If we find the character code for a space invader, we know that we have hit it, and we can POKE an explosion symbol to that location, use EXPLODE to give a suitable sound, and add a little extra to our score.

On the other hand, we may find that the memory location for the bullet's new position contains the character code for a space, 32. In this case we would just erase the bullet from its old location by poking 32 there (a space) and we would poke the character code for the bullet into its new memory location.

This process is carried out so swiftly by the Oric that it gives the impression of movement—to the naked eye the bullet will travel without pausing. In fact it is not moving at all: we are just poking the bullet to a position, then erasing it by poking a space there, and then repeating the sequence at a new position.

Often the Oric character codes will not be suitable for a particular game. On older micros there was no way around this problem, and many times I have seen a glum micro user firing a stream of full stops out of a gun which was the letter A, and trying to avoid the deadly attacks of hordes of letter Zs!

Fortunately the Oric lets us make up our own characters and use them to replace those already available. It is important to choose characters which are unlikely to be needed elsewhere. It would be very foolish to replace the number 3 by a monster character, as we would then be unable to use a 3 anywhere in the program!

Throughout this book the character codes 91-96 and 123-125 are used when new characters are created. These codes are normally used to indicate square brackets, the copyright sign, and other characters which are not likely to be needed in the course of a program.

The Oric manual goes into the methods of character creation in some detail, so we shall not repeat the explanations here. Needless to say, these programs have been written so that it is easy to insert your own characters if you find, for example, that the monsters in the program 'Zombies' are not sufficiently horrific for you.

To simplify matters, the programs all have a common format which makes them straightforward to amend.

Line 900 contains the most important screen locations. These locations are TL, the top left memory location for the screen, TR, the top right, BL, the bottom left, and BR, the bottom right. LL, the line length, or number of characters you can get on one line, is also defined.

Note that TL, TR, BL and BR are all given as hexadecimal numbers and must be preceded by #. Due to a printer quirk, the program listings have produced things like TL=£BBD2, but take no notice of the £.

There are two advantages to defining all the screen locations in one line like this. Firstly, these locations will be common to many programs, and line 900 and several other lines can be saved as a common 'skeleton' program about which the rest of the program can be constructed.

Secondly, and more important, defining the corner locations of the screen like this makes it easy to change the size and shape of the screen area to suit your own taste. You may well prefer all the action to take place in a much smaller area than that given. You do not have to use the true screen corner locations in line 900, and you may decide to move TL, TR, BL and BR in two memory locations diagonally from the actual screen corners.

You can thus make a game more or less challenging by changing just one line from the program.

The lines following 900, up to 950, similarly ease the modification of the program in other ways.

Line 910 contains the character codes for all the symbols used in the course of the game. Subroutine 2000 pokes these character codes into memory, and if you want to make up your own characters to use in a game and dispense with those given, you need only change the data statements in lines 2010 onwards.

Line 920 contains all those variables which affect the course of a game: NB might be number of bullets that you can fire, NL the number of lives you can lose before the game ends, and so on.

Again, collecting all these variables together on one line means that only minor changes are needed to make a game very easy or much more difficult.

Lines 930 and 940 are only used in certain games. They contain 8 array values, D(0) to D(7). You do not need to change these lines, but a word of explanation is necessary to clarify their function. The array contains the 8 values for movement in the 8 compass directions. To move up the screen, for example, we would need to subtract the line length from the present memory location to get the number of the location directly above it on-screen. D(0) is therefore set equal to −LL, minus the line length. The other array values give movements in the other major compass directions.

When a key is pressed on the keyboard, a variable showing the present movement number is adjusted so that it indicates which array value D( ) should be added to the present memory location to give the new position. If we pressed the key for up-screen movement, for example, the movement number variable would take the value 0, showing that D(0) is to be added to the present memory location to give the new one.

Line 950 sometimes contains DT, the distance along the screen top, and DL, the distance down the left side. Lines 960 and 965 ask whether instructions are required, and line 1000 always clears the screen and sets the foreground and background colours.

Lines 2000 and those immediately after it deal with the creation of user defined characters, and lines 5000 onwards always give the instructions and set the skill level.

Sound commands are scattered throughout the programs, and if you have delicate ears you are advised to omit these statements!

The programs usually begin with a statement that turns the cursor and keyboard click off, and then there is an immediate jump to line 900, where the initialisation of the variables begins. This may seem odd, but there are advantages to this structure. Whenever a GOTO statement is used in a program, the computer has to search through all line numbers from the start to find the line to which the jump must be made.

If we place the initialisation routines at the very start of the program and put the lines which actually run the game immediately afterwards, the game will be slowed down, because whenever a GOTO is used in the latter half of the program, the computer will have to check through all those extra program lines

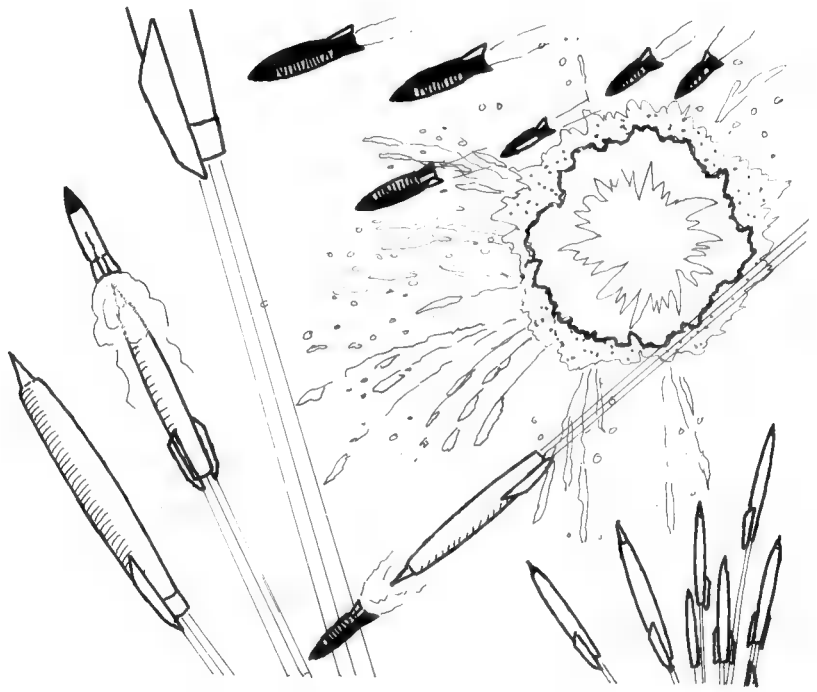
before it finds the desired one.

The programs presented here are for pleasure, and none of them is intended as an example of a perfect program. On my early model of the Oric, several of the commands did not work or appeared to have bugs in them: there are probably many improvements you can make in these programs. The concentration of all the variables in lines 900-950 should make modifications simpler.

Lastly, a word of advice. Having entered these programs, try to resist the temptation to run them immediately. Save them on cassette first. As nearly all the programs described here use POKE, a mistake on your part can cause a POKE to an area of memory off the screen. This sometimes makes the computer lose the program entirely, and you will have wasted a lot of time and effort.

Provided you have made no typing errors, the program should work perfectly. Each of these programs has been run and tested many times, and there are no major bugs. You should be able to use many of the ideas from these programs in your own games. I hope you enjoy the ones listed here.

Wallsmasher



In this game you have to stop the deadly rain of bombs by blowing them up in mid-air with your rockets. Fortunately you are protected by a thick wall of rock, but once the bombs get through that . . .

This relatively simple program illustrates many of the advantages of using only variables defined in lines 900-920. For example, you can change the number of bombs that drop at any one time by adjusting BN in line 920, or you can increase your own speed relative to the bombs by increasing BV on the same line. Find the going too tough? Increase the value of WT in line

920—this is the wall thickness. Too easy for you now? Increase TL and TR in line 900. This moves the top of the playing area down, and means there is much less time to react as the bombs drop. Changing these few variables produces many interesting variations on the game. Experiment and see what combination gives you the most challenge.

Program notes

10 Key click and cursor off.

20-40 Bomb start position routine. 20 chooses a random position along the top from which the bomb will drop. 30 ensures a bomb is not placed on top or directly above another bomb. 40 stores the start position in array B( ).

50-90 Bomb move routine. 50 is a count so that the bombs only move once for every twice the player moves. 55 checks the bomb's present position. If the bomb has been blown up leaving empty space, it is poked back into a new position at the top of the screen, 65. Otherwise the bomb is erased and its new position checked to see if the rocket is there. 75 finishes the game when any bombs hit the 'ground', and 80 pokes the remaining bombs to their new positions and saves those positions in the array B( ).

110-140 Player move routine. 110 accepts the keyboard input, which may move the rocket launcher right, 110, or left, 120. 130 prevents the launcher going off-screen at the sides, and 140 pokes it to its new position.

260-300 Player rocket launch and rocket move. 260 updates the time taken so far. In 290 the rocket launched flag RL is checked. If a rocket has been fired no other can be launched, and while a rocket is sliding through the protective wall, it should not be poked on-screen, 295.

300 prevents the player from launching a second rocket while the first is still on its way. 300-360 move the rocket, 340 causing an explosion if either a bomb or the top of the screen are hit.

370-380 gives the score.

900-950 Initialisation routine.

960-1030 Game set-up routine. 1015-1020 pokes the protective wall onto the screen, 1025 chooses the start positions for the first wave of bombs.

2000-2020 User defined character creation routine.

5000 onwards — instructions.

Important variables

BC Bomb count — bombs do not move until this reaches BV, giving the player a speed advantage.

BN Number of bombs falling at one time.

BP Bomb position on-screen.

BV Bomb velocity — a high number means the bombs move slowly.

GP Gun position — location of rocket launcher on-screen.

NP New gun position after movement.

RL Rocket launch flag — set to 1 when rocket is launched.

RP Rocket position on-screen.

Symbols used

BS Bomb symbol.

ES Explosion symbol.

GS Gun symbol for player — to launch rockets.

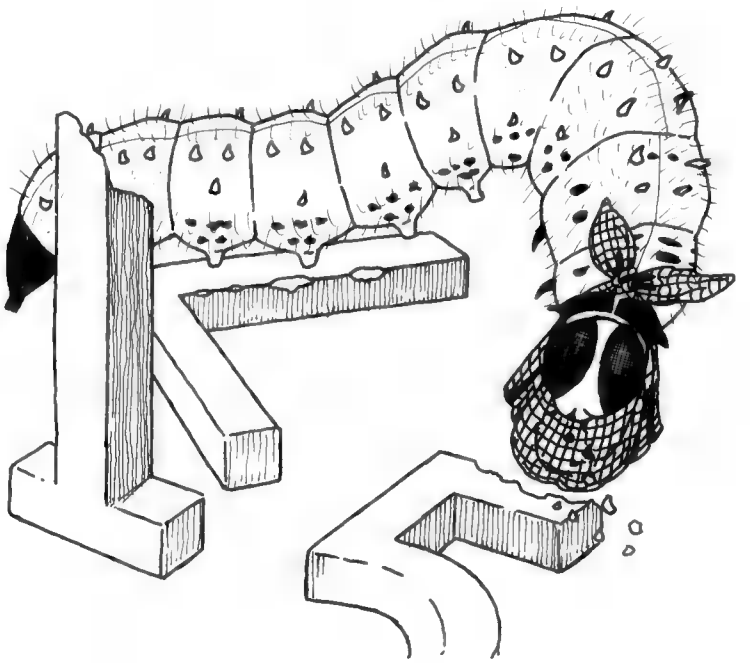
RS Rocket symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 BP=TL+INT(RND(1)*DT+1):PB=PEEK(BP):PN=PEEK(
BP+LL)
30 IFPB=BSORPN=BSTHEN20
40 B(B)=BP:RETURN
50 BC=BC+1:IFBC<BVTHEN110
55 BC=0:FORB=1TOBN:BP=B(B):PB=PEEK(BP)
60 IFPB=32THENS=S+1
65 IFPB=32THENGOSUB20
```

```
70 POKEBP,32:BP=BP+LL:PB=PEEK(BP):IFPB=RSORPB=
WSTHENPOKEBP,32:EXPLODE:GOTO90
75 IFBP>WRTHENPOKEBP-LL,ES:GOTO370
80 POKEBP,BS:B(B)=BP
90 NEXT
110 NP=GP:A$=KEY$:IFA$=CHR$(9)THENNP=NP+1
120 IFA$=CHR$(8)THENNP=NP-1
130 IFNP=GPORNP<BLORNP>BRTHEN260
140 POKEGP,32:GP=NP:POKEGP,GS
260 T=T+1:T$=STR$(T):PLOT17,24,T$
290 IFRL=1ANDPEEK(RP)=WSTHENIW=1:RP=RP-LL:GOTO
50
295 IFRL=1ANDIW=1THENRP=RP+LL:IW=0:GOTO340
300 IFRL=1THEN330
310 IFA$<>" "THEN360
320 RL=1:RP=GP-LL:GOTO290
330 PR=PEEK(RP):POKERP,32:IFPR=32ANDRP<GP-LLTH
ENRL=0:GOTO50
340 RP=RP-LL:IFPEEK(RP)=BSORRP<TRTHENPOKERP,32
:EXPLODE:RL=0:GOTO50
350 POKERP,RS
360 GOTO50
370 CLS:PRINT"You lasted for time ";T
380 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 BS=91:ES=92:GS=93:RS=94:WS=95
915 FORZ=91TO95:GOSUB2000:NEXT
920 T=0:BC=0:BN=2:BV=2:WT=2:IW=0
950 DT=TR-TL:WL=BL-LL:WR=BR-LL:GP=BL+INT(LL/2)
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
1000 CLS:INK0:PAPER3
1005 PLOT11,24,"TIME:"
1010 POKEGP,GS
1015 FORA=WLTOWL+DT:POKEA,WS:NEXTA
1020 WT=WT-1:IFWT>0THENWL=WL-LL:GOTO1015
1025 FORB=1TOBN:GOSUB20:NEXTB
1030 GOTO55
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA0,10,4,4,4,4,0,0,8,2,32,1,16,4,0,0,12
,12,12,30,30,45,45,45
2020 DATA12,12,12,12,12,63,45,45,63,33,33,33,3
3,33,33,63
```

```
5000 CLS:PRINT"You control a rocket launcher a
nd":PRINT"must try to blow up as many"
5005 PRINT"bombs as possible before you are":P
RINT"destroyed. You may only fire one"
5010 PRINT"rocket at a time - if you miss you"
:PRINT"will have to wait until the"
5020 PRINT"rocket explodes at the top of the "
:PRINT"screen before you can launch"
5030 PRINT"another. You are protected by a wal
l":PRINTWT;" bricks thick. Once this"
5040 PRINT"is breached, the game ends."
5050 PRINT:PRINT"Controls - use the cursor key
s to":PRINT"move your launcher left and"
5060 PRINT"right. Press the space bar to fire.
":PRINT"Press any key to begin."
5070 GETA$:RETURN
```

Caterpillar

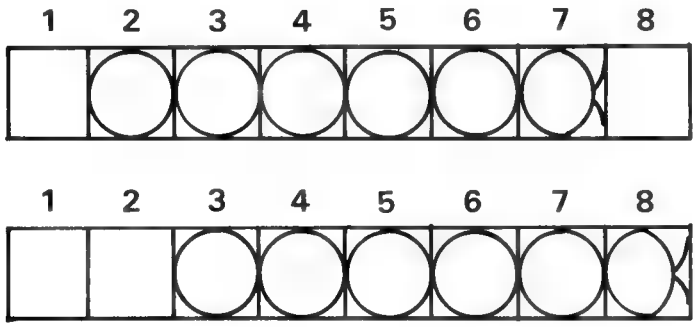


In this game you control a tiny caterpillar, and you can make it grow by gobbling up suitable food. The "food" consists of numbers which appear at random on the screen. As the game continues and your caterpillar grows longer and longer, it will become more difficult to control. But your caterpillar is so poisonous that if it bites itself, it will die. And you dare not bump into the walls either — because they're electrified!

The main problem in this program is how exactly we can make the caterpillar grow. It might seem that the simplest way to do this is to draw the caterpillar, erase it, then draw it again in its new position. When the creature eats a number, we need only

draw it one unit longer at the new position and it will appear to have grown.

There is only one thing wrong with this approach — it does not work! The Oric takes too long to draw and then erase the caterpillar. We need to devise a program which does not involve rubbing out the entire caterpillar each time. One way to do this is to only erase the very end of the caterpillar, place a segment of the body at the old position of the caterpillar's head, and draw a new head. The following diagrams should make the process clear:



Movement of the caterpillar on-screen. The last segment, at 2, is erased, the old head at 7 is replaced by a segment, and the new head is placed at 8.

The actual position at which the head will be placed depends upon which control key on the keyboard is being pressed. Because we will have to rub out the caterpillar's tail every time it moves, it is clear that we will have to store all the screen memory locations which the caterpillar occupies. Each time the caterpillar moves, the next-to-last segment will become the new tail, and we will need to know its memory location so that we can erase that new tail on the next move.

All the screen locations for the parts of the caterpillar's body are stored in the array S( ). Whenever the caterpillar moves, an unused array value is used to store the screen position where the head now is. The caterpillar gradually travels through the array, which is why the array is set to S(900) at line 900: this also gives plenty of room for it to grow in the array. When the caterpillar reaches the end of the array, the head cycles back to S(0), (line 140).

Every time the caterpillar hits a number, it grows by that number of segments, the growth taking place from the head. This allows you to control the caterpillar even as it grows.

The subroutine which places the numbers randomly on the screen has been placed at the start of the program to speed things up. It is a relatively simple task to introduce a few changes, such as symbols which will kill the caterpillar if it eats them. For example, the following have exactly this effect:

```
110 . . . . . : IF P=SS OR P=WS OR P=43 THEN 200
```

```
165 IF RND(1) >0.9 THEN N=43: GOSUB12
```

If you want to make the caterpillar's task even more difficult, change line 165 above so that the deadly symbols appear even more often, for example 165 IF RND(1)>0.1 . . . is a real challenge!

Program notes

10 Key click and cursor off.

12-16 Subroutine to place numbers on-screen.

20-60 Keyboard controls using cursor keys. The caterpillar head symbol varies according to the direction in which the caterpillar is heading, so H is adjusted along with the direction.

110-170 Caterpillar move routine. 110 finds the new head position and checks that the caterpillar has not bitten itself or run into a wall. Anything else hit must be a number, so 120 adds this value to the length of the creature, and sets the number value NV to the value of the number just hit. This is so the number can be poked back elsewhere onto the screen after it has been eaten. If the caterpillar's present length does not match its theoretical length, 130 makes the creature grow by one unit. 135 and 140 move the caterpillar through the array S( ). 145 pokes the head to its new position, pokes a body segment to the old head position, and deletes the tail. 160 pokes any eaten numbers back onto the screen by calling subroutine 12.

200-280 Game end routine. 200 makes a suitable noise when the caterpillar hits itself or the wall, and the remaining lines make appropriate comments.

900-950 Initialisation routine.

960-1000 Offer instructions and set colours.

1100-1210 Game set-up routine. 1100 and 1105 draw the electrified walls making up the playing area. 1110 to 1200 draw the caterpillar in a random position on-screen. 1210 calls the subroutine to poke the numbers 1 to 9 to the screen, and 1220 draws the caterpillar's head.

2000-2030 User-defined character creation routine.

5000 onwards — instructions.

Important variables

LP Present caterpillar length.

LS Length of caterpillar including extra units added due to numbers eaten.

MD My direction of movement — to select a move for the caterpillar from the array D( ).

NN Number position as it is poked on-screen.

NV Number value — value of number eaten.

NX Random X coordinate for number.

NY Random Y coordinate for number.

S Position of caterpillar head within array S( ).

SD Side distance inside play area.

SP Position of parts of caterpillar's body as it is poked to the screen.

ST Caterpillar tail position.

Character symbols

H Head of caterpillar. There are 4 head symbols, as the caterpillar can move in 4 different directions.

NS Nine symbol — character code for number 9.

OS Nought symbol — code for number 0.

SS Segment of caterpillar symbol.

WS Wall symbol.

Arrays

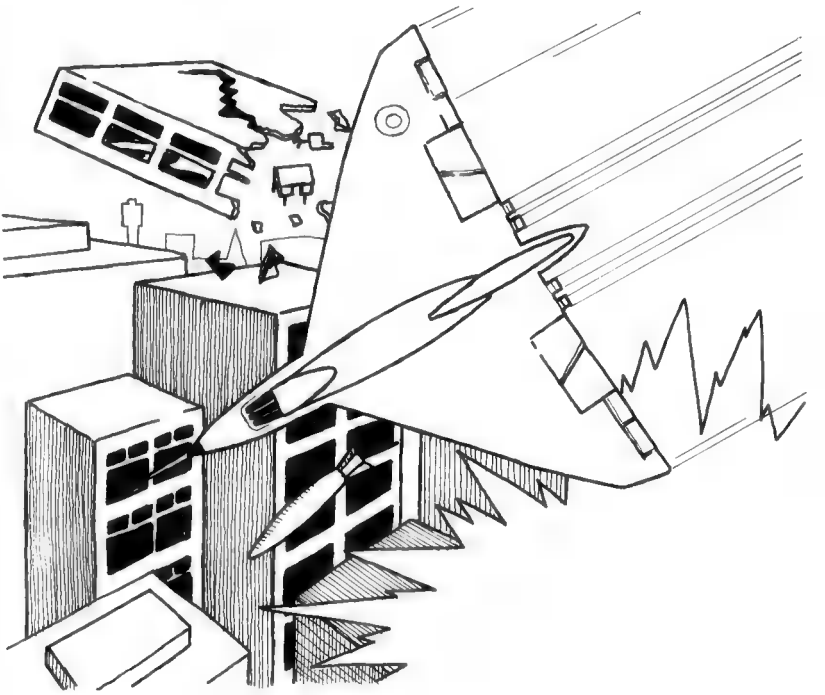
D( ) 8 compass directions from north.

S( ) Screen memory, locations of all parts of caterpilllar's body.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
12 NX=INT(RND(1)*TD+1):NY=INT(RND(1)*SD+1)
14 NN=SZ+NX-NY*LL:IFPEEK(NN)<>32THEN12
16 POKENN,N:RETURN
20 A$=KEY$:IFA$=CHR$(11)THENMD=0:H=93
40 IFA$=CHR$(9)THENMD=2:H=94
60 IFA$=CHR$(10)THENMD=4:H=95
80 IFA$=CHR$(8)THENMD=6:H=96
110 NP=SH+D(MD):P=PEEK(NP):IFP=SSORP=WSTHEN200
120 NV=0:IFP<>32THENLS=LS+P-OS:NV=P
130 IFLP<LSTHENLP=LP+1:S=S+1:PING
135 S=S+1:ST=ST+1:IFST>900THENST=0
140 IFS>900THENS=0
145 S(S)=NP:POKENP,H:POKENP-D(MD),SS:POKES(ST)
,32
160 SH=NP:IFNV>0THENN=NV:GOSUB12
170 GOTO20
200 PLAY3,0,1,1000:SOUND3,1000,0
205 POKENP,H:POKENP-D(MD),SS:POKES(ST),32:WAIT
(300)
210 CLS:PRINT"Hard luck! You ";:IFP=SSTHENPRIN
T"bit yourself.":GOTO220
215 PRINT"bumped into a wall."
220 PRINT"You grew to be ";LP;" units long."
280 END
900 DIMS(900):TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£B
EED:LL=40
910 OS=48:NS=57:SS=91:WS=92:H=95
915 FORZ=91TO96:GOSUB2000:NEXT
920 MD=4:LS=4:LP=LS:S=LS:ST=0
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:SZ=BL+2*D(1)
```

```
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK1:PAPER3
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
1110 TD=DT-4:SD=(DL-4*LL)/LL
1170 SX=INT(RND(1)*TD+1):SY=INT(RND(1)*SD+1):S
H=SZ+SX-SY*LL
1175 IFPEEK(SH)<>32THEN1170
1180 S(LS)=SH:SP=SH:POKESP,H:FORA=LS-1TO1STEP-
1
1190 NP=SP+D(RD)
1195 IFPEEK(NP)<>32THENRD=INT(RND(1)*8+1):GOTO
1190
1200 SP=NP:S(A)=NP:POKESP,SS:NEXT:FORN=0S+1TON
S:GOSUB12:NEXT
1210 SH=S(LP):GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA12,12,18,51,51,18,12,12,63,63,63,63,6
3,63,63,63
2020 DATA0,0,0,0,33,18,12,12,0,4,8,48,48,8,4,0
,12,12,18,33,0,0,0,0
2030 DATA0,8,4,3,3,4,8,0
5000 CLS:PRINT"You are a small caterpillar. Yo
u"
5005 PRINT"must try to gobble up as many":PRIN
T"numbers as you can. You will grow"
5010 PRINT"every time you eat a number.":PRINT
:PRINT"You begin a mere ";LP;" units long."
5020 PRINT"Don't bite yourself or bang your":P
RINT"head against the wall!"
5040 PRINT:PRINT"Controls - use the cursor key
s."
5050 PRINT:PRINT"Press any key to begin.":REPE
AT:GETA$:UNTILA$<>" ":RETURN
```

Bomber



In this game you pilot a bomber which flies repeatedly over a number of skyscrapers and attempts to destroy them by dropping bombs. Each hit destroys one floor of a skyscraper, so the buildings gradually get lower and lower. Unfortunately your bomber also drops lower after each successive pass, and if you fail to hit the highest buildings first the plane will run into a skyscraper and explode.

This program is an old favourite, and there are versions of it for most popular micros. The program is straightforward, but if you want to make the game harder by increasing the number of

buildings or changing their heights or positions on-screen, just amend the data values at line 2030. These values are the screen memory locations for the top of each building. Each skyscraper is drawn from the top downwards. If you have more than 10 buildings, you must change the value of NB in line 920, as this shows the number of buildings and is used by the computer to decide how many items must be read from data.

Program notes

10 Key click and cursor off.

20 Delay loop, varying according to skill level.

30-90 Bomb drop routine. Only one bomb may be dropped at a time, so 30 checks the bomb drop flag BD. If BD=1, a bomb is dropping and no further bombs can be dropped until this one explodes. If BD=0, a bomb may be dropped by pressing 'B', line 40. The bomb's first position is that of the plane, line 50. The bomb is erased, 60, and its new position calculated 65. If the new position is at 'ground' level, the bomb explodes, 70, as it does if it hits a skyscraper symbol, 80. Otherwise the bomb is just poked to its new position, 90.

100-120 Plane move routine. 100 erases the plane from its old position. If the plane has reached the far right of the screen, it is moved back to the left and placed one line lower on-screen. 110 checks if the new position is occupied already, in which case the plane will crash. 120 pokes the plane at its new position, and if the plane is not at ground level, the program continues.

130-180 Game end routine. Appropriate messages are delivered depending on whether the plane has landed or crashed. A score based on the skill level and number of building blocks destroyed is also given.

900-930 Initialisation routine.

960-1150 Game set-up routine. 1100 reads in the top position for each building, and 1105 draws it from the top down to 'ground' level.

2000-2020 User defined character creation routine.

2030 Screen memory locations for top of each building.

3000 Bomb explosion routine.

5000 onwards — instructions, including setting of skill level.

Important variables

BD Bomb drop flag — value 1 if bomb is dropping.

BP Bomb position.

HC Hit count — count for blocks of skyscraper bombed.

NB Number of buildings to be poked onto screen.

NP New plane position.

P Present plane position.

PT Plane trip count. When PT=DT (the distance from the top left to the top right of the screen), the plane has moved across the entire screen and must be moved back to the left to begin another pass.

PV Plane velocity — the higher this is set, the slower the plane moves.

TB Screen memory location for the top of each building.

Symbols used

BS Bomb symbol.

ES Explosion symbol.

PS Plane symbol.

SS Skyscraper block symbol — a building is created by stacking these on top of one another.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 WAIT(PV)
30 IFBD=1THEN60
40 A$=KEY$:IFA$<>"B"THEN100
50 BP=P:BD=1:GOTO65
60 POKEBP,32
```

```
65 BP=BP+LL
70 IFBP>BLTHENBD=0:GOSUB3000:GOTO100
80 IFPEEK(BP)=SSTHENBD=0:HC=HC+1:GOSUB3000:GOT
O100
90 POKEBP,BS
100 POKEP,32:P=P+1:PT=PT+1:IFPT=DTTHENPT=0:P=P
+LL-DT
110 NP=PEEK(P):IFNP<>32THEN150
120 POKEP,PS:IFP<BLTHEN20
130 PRINT:PRINT"Well done! You landed the plan
e":PRINT"on skill level ";SL:GOTO160
150 POKEP,ES:FORQ=1TO5:INKQ:EXPLODE:WAIT(10):N
EXT:INK0:CLS
160 PRINT:PRINT"You hit ";HC;" blocks at skill
 level ";SL
170 PRINT"A total score of ";HC*SL
180 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 BS=91:ES=92:PS=93:SS=94
915 FORZ=91TO94:GOSUB2000:NEXT
920 BD=0:HC=0:NB=10:P=TL
930 DT=TR-TL:PT=0
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK0:PAPER4
1100 FORA=1TONB:READTB
1105 FORB=TBTOBLSTEPLL:POKEB,SS:NEXTB:NEXTA
1110 FORA=BLTO£BFE0STEPLL:POKEA,18:NEXT
1150 POKEP,PS:GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA0,10,4,4,4,4,0,0,8,2,32,1,16,4,0,0,16
,8,36,63,4,8,16,0
2020 DATA45,63,45,63,45,63,45,63
2030 DATA48612,48653,48735,48776,48818,48700,4
8666,48748,48662,48627
3000 POKEBP,ES:EXPLODE:POKEBP,32:RETURN
5000 CLS:PRINT"Try to bomb all the buildings s
o":PRINT"that your plane can land"
5010 PRINT"safely. Only one bomb can be":PRINT
"dropped at a time, and no further"
5020 PRINT"bombs can be dropped until the":PRI
NT"first one explodes, so be accurate!"
```

```
5030 PRINT:PRINT"Controls - B drops bomb."
5050 PRINT:PRINT"Input skill level, 1, easy, t
o 5,":PRINT"hard";:INPUTSL
5060 IFSL<1ORSL>5THEN5050
5070 PV=6-SL:RETURN
```

Creatures

You control a man who flees from pursuing monsters by running up and down ladders and along floors. When you have time you pause to dig holes in the floor in an attempt to trap your enemies. Any creature that falls down a hole can be destroyed if you can hit it on the head while it is trapped. It will fall through the hole and splatter on the floor below. But don't take too long digging the hole or the creature will escape and crush you!

For the creatures it is important to record several pieces of information: their position on-screen; their current direction of movement; and what character normally occupies their position

on-screen, so that this can be poked back into place when the creatures next move. These values are stored in the arrays P( ), M( ), and O( ) respectively.

Holes are dug on-screen by successively poking suitable characters to the hole position to give the impression of soil being removed. At present a hole is dug in 3 stages, 210-240, but this can be reduced to a single stage by deleting 210-230 and changing 240 to POKEHP,HS:GOTO10.

Program notes

10 Keyboard click and cursor off.

15-65 Creature move routine. 15 finds the creature's present position, current movement, and the character at its old position. The last part of the line compares the creature's position to the man's and sets movement to up or down as appropriate. 20 finds the y co-ordinate of the creature relative to the man and checks if it has just fallen down a hole. If the creature is on a ladder, it will move up or down, 30. 40 finds its new position and checks that it is not moving to an occupied position. 50 fills in the hole if the creature has just been knocked through. 60 moves the creature to its new position.

70-190 Move man routine. 70-120 are keyboard controls. 130 finds the new position. 150 makes the man fall vertically if he has jumped through a hole, 160 prevents him from going off the end of a floor. 170 moves the man and 180 finds his y co-ordinate relative to BL: this is needed to determine the direction of creature movement. 190 ends the game if the man gets eaten.

200-240 Hole digging routine. 200 prevents a hole being dug at 'ground' level or on top of a ladder, 210-240 substitute different characters as the hole is dug.

300-340 Falling creature routine. 300 repairs the floor behind the falling creature, 305-310 make it fall until it hits the floor below. 315 explodes the body, and 320-340 poke a new creature to an unoccupied position.

500-510 Game end routine.

900-955 Initialisation routine.

960-1130 Game set-up routine. 1010-1025 draws each successive floor and, except for the ground floor, 1030-1070 draws randomly-placed ladders hanging down from that floor to subsequent ones. 1080 places the man, and 1085-1130 place the creatures at random positions on a floor other than the man's.

2000-2030 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

CM Creature movement direction.

CN Creature's new position.

CO Character at creature's old position.

CP Present creature position.

CV Creature value — its worth to player.

DF Vertical screen distance between floors.

EC Escape chance — probability of creature caught in hole escaping.

F Flag for man falling through hole to new floor.

G Creature movement on ladder, if one is available.

H Half, 0.5, used in random number calculations.

HM Hole movement — direction in which hole will be dug by man.

HP Hole position.

LB Ladder base position.

LD Horizontal distance between ladders on a floor.

LT Ladder top position.

M Minus one, −1.

MM Man's movement.

MP Man's position.

NC Number of creatures in game.

NF Number of floors.

NH Number of digs to produce complete hole.

NL Number of ladders between pairs of floors.

NP New position for man.

P Plus one, +1.

SC Player score.

Z Zero, 0.

Symbols used

BS Blank space.

CS Creature symbol.

ES Explosion symbol.

FS Floor symbol.

HS Hole symbol — for completed hole.

H1, H2, H3 Holes in various stages of completion.

LS Ladder symbol.

MS Man symbol.

OS Old symbol — found at man's last position.

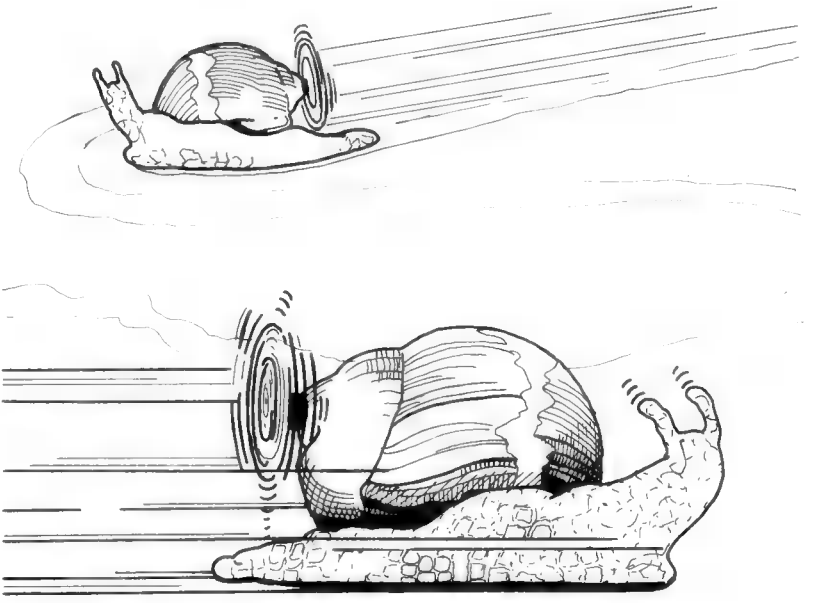
```
10 PRINTCHR$(6)CHR$(17):GOTO900
15 FORA=PTONC:CP=P(A):CM=M(A):CO=O(A):G=LL:IFM
P-CP<ZTHENG=-G
20 C=BL-CP:Y=INT(R+C/LL)-MY:IFCO=HSANDPEEK(CP)
=HSTHEN300
25 W=PEEK(CP+CM):IFW=BSORW=CSTHENM(A)=-CM
30 IFCO=LSANDY<>ZANDPEEK(CP+G)<>BSTHENCM=G
40 CN=CP+CM:PC=PEEK(CN):IFPC=CSORPC=BSORCO=HSA
NDRND(1)>ECTHEN65
50 IFCO=HSTHENCO=FS
60 POKECP,CO:POKECN,CS:P(A)=CN:O(A)=PC:IFPC=MS
THEN500
65 NEXTA
```

```
70 HM=Z:A$=KEY$:IFA$=CHR$(11)THENMM=-LL
80 IFA$=CHR$(10)ORE=PTHENMM=LL
90 IFA$=CHR$(8)THENMM=M
100 IFA$=CHR$(9)THENMM=P
110 IFA$=","THENHM=M:GOTO200
120 IFA$="."THENHM=P:GOTO200
130 NP=MP+MM:PN=PEEK(NP):IFMP=NPTHEN15
140 IFF=PANDOS<>BSTHENF=Z
150 IFOS=HSTHENF=P:MM=LL:NP=MP+LL:PN=BS
160 IFPN=BSANDF=ZORPN=MSTHEN15
165 PLAY7,0,1,250:PLAY7,0,2,250
170 POKEMP,OS:OS=PN:MP=NP:POKEMP,MS
180 J=BL-MP:MY=INT(R+J/LL):IFPN<>CSTHEN15
190 GOTO500
200 HP=MP+HM:H=PEEK(HP):IFHP>=BLORH=LSTHEN15
205 PLAY0,7,1,250
210 S=HS:IFH=FSTHENS=H1
220 IFH=H1THENS=H2
230 IFH=H2THENS=H3
240 POKEHP,S:GOTO15
300 POKECP,FS:PLAY1,0,2,10:SO=0
305 CP=CP+LL:E=PEEK(CP):FORV=SOTOSO+20:SOUND1,
V,4:NEXT:SO=SO+20
310 IFE=BSTHENPOKECP,CS:WAIT(T):POKECP,BS:GOTO
305
315 PLAY0,0,0,0:POKECP,ES:EXPLODE:SC=SC+CV
320 CO=E:CM=P:CN=TL+DF*INT(RND(1)*NF)+1:IFRND(
1)>HTHENCN=CN+DT:CM=M
330 PC=PEEK(CN):IFPC=CSORPC=BSTHEN320
340 M(A)=CM:GOTO50
500 PLAY7,0,3,1:WAIT(40):PLAY0,0,0,0:WAIT(200)
505 CLS:PRINT"Your score at skill level ";SL;"
 was ";SC
510 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 LS=91:FS=92:MS=93:CS=94:HS=95:ES=96:OS=FS:
BS=32:H1=123:H2=124:H3=125
915 FORZ=91TO96:GOSUB2000:NEXT:FORZ=123TO125:G
OSUB2000:NEXT
920 SC=0:NC=4:NF=4:NH=4:NL=3:P=1:Z=0:M=-1:H=.5
:T=10
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=(BL-TL)/LL:DF=INT(H+DL/(NF-1))
*LL
```

```
955 LD=INT(DT/NL):R=(LL-1)/LL
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK4:PAPER3
1010 B=TL+1
1015 D=B+DT:FORA=BTOD:IFPEEK(A)=LSTHEN1025
1020 POKEA,FS
1025 NEXTA:IFB>=BLTHEN1080
1030 FORA=1TONL:IFA>1THENLT=LT+LD:GOTO1040
1035 LT=B+INT(RND(1)*LD+1)
1040 IFPEEK(LT)=LSTHEN1035
1045 LB=LT+DF:IFLB>D+DFTHENLB=D+DF
1050 FORD=LTTOLBSTEPLL:POKED,LS:NEXTD:NEXTA
1070 B=B+DF:GOTO1015
1080 MP=TL:OS=PEEK(MP):PN=PEEK(MP+1):POKEMP,MS
1085 FORA=1TONC
1090 CP=MP+DF+INT(RND(1)*DT+1)+INT(RND(1)*(NF-
1))*DF
1100 IFPEEK(CP)<>FSTHEN1090
1110 CM=P:IFRND(1)>HTHENCM=M
1130 P(A)=CP:M(A)=CM:O(A)=FS:POKECP,CS:NEXTA:G
OTO180
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA33,33,63,33,33,33,63,33,63,63,63,63,6
3,63,63,63,14,14,4,31,4,4,10
2020 DATA17,63,12,45,45,0,18,0,45,1,1,1,1,1,1,
1,1,8,2,32,1,16,4,0,0
2030 DATA1,1,1,63,63,63,63,63,1,1,1,1,1,63,63,
63,1,1,1,1,1,1,1,63
5000 CLS:PRINT"You must try to kill all the":P
RINT"creatures by digging holes for"
5005 PRINT"them and then hitting them over the
":PRINT"head while they are trapped in"
5010 PRINT"the hole. If you leave it too long"
:PRINT"the creature will escape and"
5015 PRINT"eat you. Use the cursor keys ":PRIN
T"to move, and the < and > keys"
5020 PRINT"to dig holes to the right and left"
:PRINT"respectively. It takes ";NH
5025 PRINT"digs to complete a hole - a hole":P
RINT"that has only partially been dug"
5030 PRINT"will not stop the creatures.":PRINT
"You may jump through a hole yourself"
```

```
5035 PRINT"and you will fall unharmed to the":
PRINT"floor below."
5050 PRINT"Input your skill level, 1 to 9 the"
:PRINT"higher numbers mean that the"
5055 PRINT"creatures escape more quickly from"
:PRINT"the holes, but they are worth"
5060 PRINT"more";:INPUTSL
5070 EC=SL/10:CV=SL:RETURN
```

Snailtrail



In this program you control a snail which moves around the screen at a very un-snailish speed. Every time your snail moves it leaves a slimy trail behind it — and you must not bump into the trail!

Seem easy enough? Well, there is one small problem: the computer also controls a snail, so not only must you avoid hitting your own trail, but also that left by the computer's snail. In addition you must not hit the other snail, or the walls . . . .

The program is comparatively straightforward, the main problem being to arrange for the computer's snail to move in a random fashion which, at the same time, is not so unpredictable that it becomes easy to trap the snail in a small portion of the screen.

In early versions of the program the computer's snail moved completely at random. This proved self-defeating as the snail boxed itself in and was forced to eat its own trail.

In this version the snail only changes direction around 3% of the time, line 135, although it will always try to change direction if it is about to hit a trail, the other snail, or the wall.

If a direction change is called for, the program chooses a random direction, line 140, and if there is no obstacle in that direction, this is the way the snail will move, 145. Otherwise the program tests all other movements from the present position in an attempt to find a 'safe' direction. This is the reason for lines 140 and 150: REPEAT try all the directions UNTIL you've tried them all or a safe direction has been found.

Program notes

10 Keyclick and cursor off.

15 Loop for a series of games.

20-80 Keyboard controls to direct the snail.

110 Check new position of snail. If it is occupied, game to the computer!

120 Move snail to new position.

125 Pause of varying length at different skill levels.

130 Check computer snail's new position if it carries on in its present direction.

135-150 If the new position is empty, move to it. The RND statement ensures that the snail will still change direction occasionally even when the space in front is empty.

160 Check computer snail's revised position. If it is empty, move the snail and the game continues.

170-190 Otherwise blow up the computer's snail and start another game.

200-220 The score after a series of games.

900-950 Initialisation routine.

980-1000 Give instructions, set foreground and background colour, etc.

2000-2020 User defined character creation routine.

3000-3010 Draw walls.

3020-3040 Place snails at their start positions and display score.

5000 Instructions and choice of skill level.

Important variables

B Count for number of directions tried by computer's snail in its attempt to find a safe movement.

CC Computer count — score for computer in terms of games won.

CD Computer direction for snail.

CP Computer position for snail.

D Direction randomly selected by computer's snail.

M Movement direction for snail at present.

MC My count — my score in terms of games won.

MD My direction for snail.

MP My position for snail.

NC New computer position for snail.

NG Number of games to be played together.

NP New position for my snail.

P Peeked value found at new positions of snails.

Q Count for games as they are played.

SL Skill level.

Symbols used

CS Computer snail symbol.

ES Explosion symbol — used when a snail hits something.

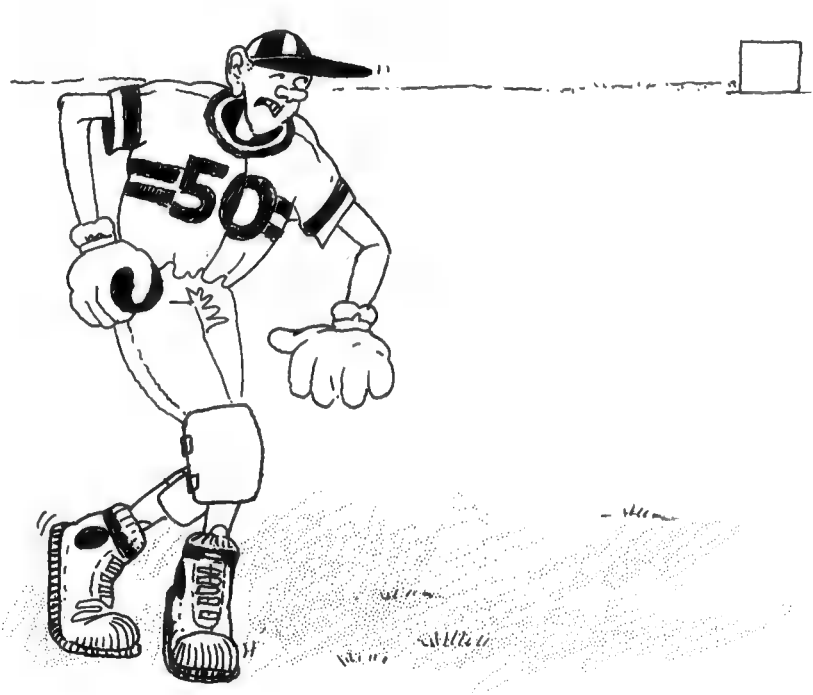
MS My snail symbol.

WS Wall symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
15 FORQ=1TONG:GOSUB3000
20 A$=KEY$:IFA$=CHR$(11)THENMD=0
40 IFA$=CHR$(9)THENMD=2
60 IFA$=CHR$(10)THENMD=4
80 IFA$=CHR$(8)THENMD=6
110 NP=MP+D(MD):P=PEEK(NP):IFP<>32THENPOKENP,E
S:CC=CC+1:GOTO180
120 MP=NP:POKEMP,MS
125 WAIT(11-SL)
130 NC=CP+D(CD):C=PEEK(NC)
135 IFC=32ANDRND(1)>.03THENCP=NC:POKECP,CS:GOT
O20
140 M=CD:B=0:D=2*INT(RND(1)*4):REPEAT:B=B+1:D=
D+2:IFD>6THEND=0
145 IFPEEK(CP+D(D))=32THENM=D
150 UNTILB=4ORM<>CD
160 NC=CP+D(M):C=PEEK(NC):IFC=32THENCP=NC:CD=M
:POKECP,CS:GOTO20
170 POKECP,CS:MC=MC+1
180 EXPLODE:WAIT(200)
190 NEXT
200 CLS:PRINT"The final score was:":PRINTSPC(1
3);"Computer : ";CC
210 PRINTSPC(18);"You : ";MC
220 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 MS=91:CS=92:WS=93:ES=94
915 FORZ=91TO94:GOSUB2000:NEXT
920 MC=0:CC=0:MD=0:CD=2
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:TH=INT(DT/3)
980 GOSUB5000
1000 CLS:INK1:PAPER3:GOTO15
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA12,12,18,51,51,18,12,12,12,30,30,63,6
3,30,30,12
2020 DATA63,63,63,63,63,63,63,63,8,2,32,1,16,4
,0,0
3000 CLS:FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
3010 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
```

```
3020 MF=TL+TH+10*LL:CP=TR-TH+10*LL:MD=0:CD=2
3025 C$="Computer : "+STR$(CC):M$="You : "+STR
$(MC)
3030 PLOT6,24,C$:PLOT25,24,M$
3035 PLAY0,1,4,100
3040 POKEMP,MS:POKECP,CS:RETURN
5000 CLS:PRINT"You will leave a track behind":
PRINT"you as you move. The computer"
5005 PRINT"will also leave a different track."
:PRINT"The first one to bump into a"
5010 PRINT"track loses the game.":PRINT:PRINT"
How many games do you want";:INPUTNG
5015 PRINT:PRINT"What skill level, 1, slow, to
":PRINT"10, fast";:INPUTSL
5020 IFSL<1ORSL>10THEN5015
5040 PRINT:PRINT"Controls - use the cursor key
s."
5050 PRINT:PRINT"Press any key to begin.":REPE
AT:GETA$:UNTILA$<>" ":RETURN
```

Wipeout



This game requires the player to try and wipe out a screen full of targets by hitting them with a ball. The game has several useful routines which can easily be incorporated into other programs.

A few years ago one of the early popular arcade games was Breakout, where the player scored by knocking bricks out of a wall. The game that follows is not a Breakout variation, but Breakout can easily be programmed using the routines given here for bouncing the ball off the walls of the playing area.

The big problem with this type of game is that writing the program to use low resolution graphics makes it impossible for

the ball to bounce at a great variety of angles. To enable the player to make different shots, the player's bat consists of 3 symbols, and the ball bounces in a variety of ways depending where on the bat it strikes.

Program notes

10 Keyboard click and cursor off.

20-30 Keyboard controls to move bat left, 20, to stop it, 25, or to move it right, 30.

40-50 Bat move routine. 40 finds the new middle position of the bat after movement, and checks that the bat will not be off-screen or on top of the ball. 50 moves the bat to its new position.

55 Score – number of crosses hit so far.

60-180 Ball move routine. 60 finds the new ball position. 65 checks if this is a cross, in which case the score is incremented. If the new position is empty or a cross, the ball can be moved to that position with no difficulty, 70. Complications arise when the ball hits a wall or the bat, as it must bounce in a different direction. 80 takes care of bounces off the top wall, and 90 does a similar job for side wall bounces. 110 is needed in case the ball is about to hit the bottom left or bottom right of the walls. Whereas normally the ball will be reflected when it strikes a wall, doing so at the very bottom edges sends the ball out of the playing area on-screen and into free memory! So if a ball is about to strike BL or BR, its movement is just reversed and it travels back along its path. The beginning of 110 can be omitted: this causes an occasional reversal of movement as the ball bounces from wall or bat. 120 and 130 determine the new ball movement if the bat is struck, 130 deflecting the ball slightly if the right or left sides of the bat have been struck. Having done all this, it is possible that the ball has just been deflected from one wall onto another, so 140 returns to the start of the ball move routine to check that the new position is clear. 150 erases the ball from its old position, and if the new position is not off the bottom of the playing area due to the bat missing the ball, the ball is drawn to its new position and the process begins again. 160 checks if all the crosses have been hit, 175 increases the ball count if the ball has gone off screen, and 180

pauses before a new ball comes into play, 185 chooses a random start position near the bottom of the playing area to poke the new ball into play.

190-210 Score and comments at end of game.

900-950 Initialisation routine.

960-1150 Game set-up routine. 1100 draws the top wall. 1105 the side walls. 1110-1130 poke the target crosses into position on screen. 1110 sets up the start and end position of each row of crosses, EX being the amount added on so that alternate rows are staggered. 1120 counts the crosses, pokes them into place. 1130 works out the start and end positions for individual rows of crosses. Finally, 1140 pokes the bat into position in the middle at the bottom of the playing area.

2000-2020 User defined character creation routine.

5000 onwards — instructions.

Important variables

BC Ball count — how many balls used so far.

BM Ball movement.

BT Total number of balls allowed — when BC reaches this value, the game ends.

CH Number of crosses hit by ball.

CT Total number of crosses.

DL Distance along left of screen.

DT Distance along top of screen.

EP End position for poking crosses on screen.

EX Extra added to stagger rows of crosses.

L2 Double line length LL.

M Bat movement.

MP Position of middle of bat.

NB New ball position.

NP New bat position.

SP Start position for poking crosses on screen.

Symbols used

BS Ball symbol.

CS Cross symbol.

LS Left bat end symbol.

MS Middle of bat symbol.

RS Right bat end symbol.

SS Side wall symbol.

WS Top wall symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 A$=KEY$:IFA$=CHR$(8)THENM=-1
25 IFA$=" "THENM=0
30 IFA$=CHR$(9)THENM=1
40 NP=MP+M:IFNP<=BLORNP>=BRORPEEK(NP+M)=BSORNP
=MPTHEN55
50 POKEMP-M,32:MP=NP:POKEMP-1,LS:POKEMP,MS:POK
EMP+1,RS
55 PLOT27,24,STR$(CH)
60 NB=BP+BM
65 PB=PEEK(NB):IFPB=CSTHENCH=CH+1
70 IFPB=32ORPB=CSTHEN150
80 IFPB=WSTHENBM=BM+L2:PLAY3,0,1,100:SOUND3,10
0,0:GOTO60
90 IFPB=SSTHENBM=L2*(BM/ABS(BM))-BM:PLAY3,0,1,
100:SOUND3,100,0:GOTO60
110 IFRND(1)<.05ORNB=BLORNB=BRTHENBM=-BM:GOTO6
0
120 PING:BM=BM-L2:IFPB=MSTHEN60
130 NB=BP+BM+1:IFPB=LSTHENNB=NB-2
140 GOTO65
150 POKEBP,32:BP=NB:IFBP<BLANDCH<CTTHENPOKEBP,
BS:GOTO20
160 IFCH=CTTHEN190
170 PLAY7,0,1,1000:FORQ=1TO10:SOUND3,Q,0:NEXT
```

```
175 BC=BC+1:IFBC>BTTHEN190
180 WAIT(200):BM=-LL+1:IFRND(1)>.5THENBM=BM-2
185 PLOT12,24,STR$(BC):BP=(BL-LL)+INT(RND(1)*(
DT-3)+2):POKEBP,BS:GOTO20
190 WAIT(500):CLS:IFCH=CTTHENPRINT"You got the
m all!":END
200 PRINT"You hit ";CH;" crosses altogether."
210 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 CS=43:BS=91:LS=92:MS=93:RS=94:SS=95:WS=96
915 FORZ=91TO96:GOSUB2000:NEXT
920 CH=0:BC=0:BT=3:CT=0
950 DT=TR-TL:DL=BL-TL:L2=LL*2:SP=TL+LL+1:EP=BL
-L2+1
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK1:PAPER7
1100 FORA=TLTOTR:POKEA,WS:NEXTA
1105 FORA=TLTOBL-LLSTEPLL:POKEA,SS:POKEA+DT,SS
:NEXTA
1110 A=SP:B=SP+DT:EX=-1
1120 CT=CT+1:POKEA,CS:A=A+2:IFPEEK(A)=32ANDPEE
K(A-1)=32THEN1120
1130 EX=-EX:A=SP+EX+L2:SP=A:B=A+DT:IFA<EPTHEN1
120
1140 MP=BL+INT(DT/2):POKEMP-1,LS:POKEMP,MS:POK
EMP+1,RS
1150 PLOT5,24,"BALL:":PLOT12,24,STR$(BC):PLOT1
9,24,"SCORE:":GOTO175
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA0,12,18,33,33,18,12,0,63,56,48,32,0,0
,0,0,63,0,0,0,0,0,0,0
2020 DATA63,7,3,1,0,0,0,0,51,63,63,63,63,63,63
,63,63,63,63,63,63,63,63
5000 CLS:PRINT"In Wipeout you must try to wipe
":PRINT"out as many targets as you can"
5005 PRINT"by bouncing a ball onto them. Your"
:PRINT"bat moves along the bottom of"
5010 PRINT"the screen - its left side will":PR
INT"deflect a ball to the left and its"
5020 PRINT"right side deflects to the right.":
PRINT"Any ball missed is lost, and you"
```

```
5025 PRINT"have ";BT;" balls to try and hit":P
RINT"all the targets with."
5050 PRINT:PRINT"Controls - use the cursor key
s to":PRINT"move the bat left or right."
5055 PRINT"Press the space bar to stop the bat
":PRINT"from moving."
5060 PRINT"Press any key to begin.":REPEAT:A$=
KEY$:UNTILA$<>"":RETURN
```

Zombies

This program turns up in many guises: the idea is that the player seeks to evade some extremely stupid pursuers by luring them into traps. In some variations, the pursuers are robots, destroyed if they strike electrified posts scattered randomly around. Here the enemies are blind, flesh-eating zombies, who can be eliminated by making them fall down potholes. Another version has the player as a swimmer fleeing sharks. The latter die if they hit each other or any of the jellyfish that float around. The pattern is clearly the same, so if the zombies do not appeal, define a few characters of your own and change the program to one of the other versions.

Zombies is an enjoyable game to play, as it can either be played as a game of skill, or, with minor amendments, as a game of fast reflexes. As set up at present, the zombies will only move after the player has made a move by inputting a value at the keyboard, lines 20-100. Changing the GETA$ in 20 to A$=KEY$ and eliminating the REPEAT in 20 and deleting 100 means that the zombies will now move without waiting for a keyboard input. You will need to add: 105 IFMD>7THEN140.

One thing that confuses many people in this game is the zombie movement. Zombies always move towards you. This does not necessarily mean that they move in the same direction as you have just moved. They move so as to minimise the distance between you and them. This sometimes involves moving in the same direction, but on other occasions the zombie will move differently. After you have played a few games it will become easier to trap zombies in potholes as you will be able to tell in which direction they are going to move. Then is the time to try a higher skill level . . . or move on to the real-time game!

Program notes

10 Keyboard click and cursor off.

20-100 Keyboard controls.

115-130 Player move routine. 110 erases the player's symbol from its old position and finds the new position. 115 checks if this position is a pothole, and 120 checks if it is a zombie. 130 pokes the player's symbol to the new position.

140-255 Zombie move routine. 140 finds the player's co-ordinates relative to BL. 150 picks out each zombie position from the array Z( ), and checks if that position is empty or a pothole. In either case the zombie has been killed and can no longer move. 160 finds the zombies' co-ordinates relative to BL, and these co-ordinates are compared to those for the player so that the program can decide which x and y axis movements the zombies should make to bring them closer to the player, lines 170-200. 210 erases the zombie from its old position and examines the new position. 220 checks for a pothole, 230 checks if the new position is that of the player, and 240 ensures that if there is already a zombie at the new position, the other zombie does not move on top of it. 250 pokes

the zombie to its new position, and 255 completes the loop.

260-310 Game end routine. 260 checks if all the zombies are dead. If not, provided the player is not dead, the game continues, 270. 275-310 comment on your fate and give the score.

900-950 Initialisation routine.

960-1190 Game set-up routine. 1100 and 1105 put potholes all around the screen edge. 1120-1145 poke potholes or zombies at positions depending on the outcome of RND. 1150 chooses a random position on the main diagonal for the player's symbol, 1155 ensures there is not a zombie at that position, and 1160-1180 clear the surrounding area of the beasts. 1190 pokes the player's symbol on-screen and the game begins.

2000-2010 User defined character creation routine.

3000-3020 Routine used when player falls down pothole.

5000 onwards — instructions and setting of skill level.

Important variables

CP Chance of pothole as these are poked at start.

CZ Chance of zombie as these are poked at start.

EP End position as potholes and zombies are poked to the screen.

MD My direction — used to pick movement from array D( ).

MK Man killed flag — set to 1 to show death.

MP My position.

MX My x coordinate relative to BL.

MY My y coordinate relative to BL.

MZ Maximum number of zombies at particular skill level.

NP My new position.

NZ Number of zombies present in game.

R Random number less than 1.

SL Skill level.

SP Start position for poking potholes, zombies.

XM Z movement for zombie.

YM Y movement for zombie.

ZK Number of zombies killed.

ZN New zombie position.

ZP Present zombie position.

ZX Zombie x coordinate relative to BL.

ZY Zombie y coordinate relative to BL.

Symbols used

MS My symbol.

PS Pothole symbol.

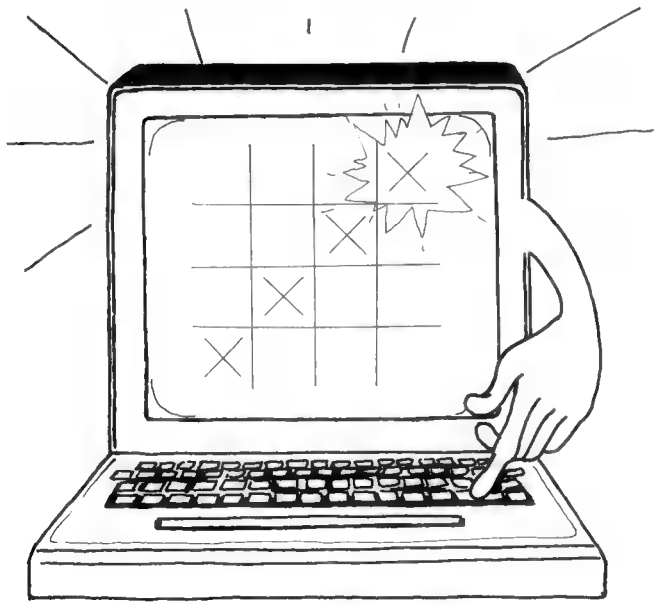
ZS Zombie symbol.

```
5 REM ZOMBIES
10 PRINTCHR$(6)CHR$(17):GOTO900
20 REPEAT:MD=8:GETA$:IFA$="W"THENMD=0
30 IFA$="E"THENMD=1
40 IFA$="D"THENMD=2
50 IFA$="C"THENMD=3
60 IFA$="X"THENMD=4
70 IFA$="Z"THENMD=5
80 IFA$="A"THENMD=6
90 IFA$="Q"THENMD=7
100 UNTILMD<8
110 POKEMP,32:NP=MP+D(MD):P=PEEK(NP)
115 IFP=PSTHENGOSUB3000:GOTO290
120 IFP=ZSTHEN280
130 MP=NP:POKEMP,MS
140 MY=INT((BL-MP)/LL)+1:MX=MP-BL+MY*LL
150 FORZ=1TONZ:ZP=Z(Z):PZ=PEEK(ZP):IFPZ=32ORPZ
=PSTHEN255
160 ZY=INT((BL-ZP)/LL)+1:ZX=ZP-BL+ZY*LL
170 XM=0:IFZX<MXTHENXM=1
180 IFZX>MXTHENXM=-1
190 YM=0:IFZY<MYTHENYM=-LL
```

```
200 IFZY>MYTHENYM=LL
210 ZN=ZP+XM+YM:P=PEEK(ZN):POKEZP,32
220 IFP=PSTHENZK=ZK+1:Z(Z)=ZN:GOTO255
230 IFP=MSTHENMK=1
240 IFP=ZSTHENZN=ZP
245 SOUND1,Z*10,0:PLAY1,0,4,(ABS(ZX-MX))
250 Z(Z)=ZN:POKEZN,ZS
255 NEXT
260 IFZK=NZTHENPLOT2,24,"Well done! There are
no zombies left!":GOTO285
265 S$="Zombies killed = "+STR$(ZK):PLOT8,24,S
$
270 PLAY0,0,0,0:IFMK=0THEN20
275 PLOT5,24,"The zombie crunches you up.":SOU
ND5,100,0:PLAY0,1,4,100
280 WAIT(200):PLAY0,0,0,0
285 WAIT(600)
290 CLS:PRINT"You killed ";ZK;" zombies out of
 ";NZ
300 PRINT"This is a success rate of ";INT(100*
ZK/NZ);" percent"
305 PRINT"at skill level ";SL
310 END
900 DIMZ(30):TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BE
EC:LL=40
910 MS=94:PS=95:ZS=96
915 Z=MS:GOSUB2000:Z=PS:GOSUB2000:Z=ZS:GOSUB20
00
920 ZK=0:MK=0:MZ=30:CP=.1
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:SP=TL+LL+1:EP=BR-LL-1:B=
SP
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK4:PAPER3
1100 FORA=TLTOTR:POKEA,PS:POKEA+DL,PS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,PS:POKEA+DT,PS:NE
XT
1120 FORA=BTOB+DT-2:R=RND(1):IFR<CPTHENPOKEA,P
S
1130 IFR<CZANDR>CPANDNZ<MZTHENNZ=NZ+1:Z(NZ)=A:
POKEA,ZS
```

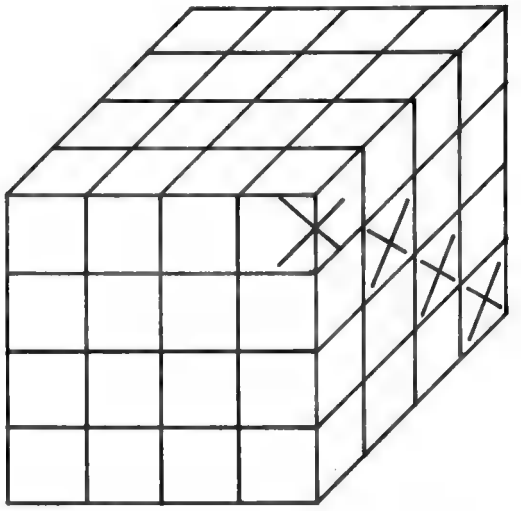
```
1145 NEXT:B=B+LL:IFB<EPTHEN1120
1150 MP=SP+D(3)*INT(RND(1)*(15)+1)
1155 IFPEEK(MP)=ZSTHEN1150
1160 FORD=0TO7:PP=MP+D(MD):IFPEEK(PP)=ZSTHENZK
=ZK+1
1180 POKEPP,32:NEXT
1190 POKEMP,MS:GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA14,14,4,31,4,4,10,17,12,18,33,33,33,1
8,12,0,4,14,21,31,17,31,21,21
3000 PLOT8,24,"You fall into a pothole.    "
3010 FORQ=1TO100:SOUND1,Q,4:NEXT:EXPLODE
3020 PLAY0,0,0,0:WAIT(500):RETURN
5000 CLS:PRINT"You are trapped on Zombie Islan
d":PRINT"which is full of deep potholes."
5005 PRINT"The zombies on the island want to":
PRINT"eat you, and although they are"
5010 PRINT"blind when you move they will hear"
:PRINT"you and rush towards you. If "
5015 PRINT"are careful you can make the ":PRIN
T"zombies fall into the potholes. If"
5020 PRINT"you fail you will be eaten.":PRINT:
PRINT"Controls - use the keys around the"
5025 PRINT"S key to move.":PRINT:PRINTSPC(16);
"Q W E":PRINTSPC(17);"A S D":PRINTSPC(18);"Z X
 C"
5050 PRINT:PRINT"Input your skill level, 1, ea
sy, to 10, difficult";
5060 INPUTSL:IFSL<1ORSL>10THEN5050
5070 CZ=CP+SL*.008:RETURN
```

Artificial Intelligence Program



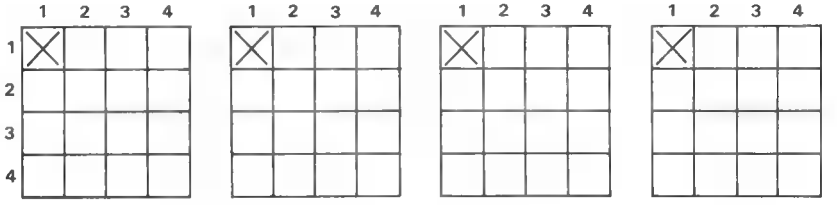
One of the great pleasures of computing is that the computer can be a ready made opponent in any number of different games. It is a real challenge to develop a program which enables the machine to play a game in an intelligent manner. The real test is if the program is good enough to beat its creator, and the program that follows has passed that test many a time!

In essence the game is the familiar noughts and crosses, but it is made much more challenging by being played on a 4 by 4 by 4 cube, the object being to get a straight line of four Os or four Xs in any direction:



A winning line when playing noughts and crosses on a 4 by 4 by 4 cube.

It would obviously be difficult to show the cube on-screen, and even more difficult to show the moves clearly, so the cube is 'sliced' into four sections, and these are displayed side by side. It is important to remember when playing the game that these sections are actually on top of each other, and the following diagram gives an example of a line of Xs which goes through all four sections:



Representation of the cube on-screen, showing a winning line of Xs.

If this is hard to understand, try to imagine the four sections piled on top of each other from left to right: the Xs then form a vertical line through the cube.

The method by which the program chooses its move is a complex one, but in outline the procedure is as follows.

You make a move. The computer finds all the possible straight lines passing through the position where you have just moved. It then examines all those lines to find out the situation in each line: for example, your move might have created XXX. in one line and OX.O in another.

Every type of line has been given a value at the start of the program. XXX. would have a high value, because unless the computer stops you, you are going to win next move! The computer keeps a running total stored for every position in the cube, and it now adds the line value to the running totals for every position in the line. The running totals for all the positions which make up XXX. would now be very high, because a large value would have been added to each of them. On the other hand, the computer would rate a line like OX.O as one of little value, because neither side can make a line of four on that line. Consequently, the computer would only add a small amount to the running totals for each of these positions.

Once the computer has found all the lines affected by your move and changed all the running totals it is now ready to make its own move. It does this by looking through the running totals for every position on the cube, and picking for its move the poition with the highest running total. Any position with a high total must be on several valuable lines, for example O.OO or XXX. or perhaps at a point where two lines like OO.. cross and a single move will make both lines into OOO.

Basically, the computer chooses as its move the position which at the same time lies on several lines where it has already placed a O, and/or also lies on lines which contain several Xs. The program can be beaten, but you'll have to play very well to do it!

Program notes

10 Keyboard click and cursor off.

20-100 Move input routine. 70 allows you to delete a move if you change your mind, 80 checks that the move is legal.

105-150 Computer move routine. The move is selected as described above.

130 counts the moves in twos (for one O and one X), and 140

checks if move 64 has been reached, in which case the game is drawn.

200-320 Board draw routines. Each position in the cube is stored in the array C(A,B,C), positions with a value of 1 being empty, 2 being O, and 3 being X. 300-320 checks each of the array values and plots a ., O, or X in the appropriate position.

400-540 Line check routine. This finds which of the 13 lines which pass through the cube the move position lies on.

600-630 Highest total routine. The computer examines the running total for every position in the cube, and saves the highest total in H and the coordinates of that position in A1, B1 and C1, line 620. These become the coordinates of the computer's move, 630.

700-950 Line value routine. Once a line has been found in routine 400, the computer works out the situation on that line and adds an appropriate value to the running total for every position on the line.

1000-1050 Valuation routine. Once the type of line has been found, this supplies a suitable valuation of the situation in that line.

1100-1210 Congratulatory messages for the winner.

1800-1820 Erases abandoned or illegal moves.

1900-1980 Initialisation routine.

5000 onwards — instructions.

Important variables

A$ Player's move as a string.

C( ) Information on all positions within cube: values of 1 indicate an empty position, 2, a O, and 3, a X.

CO Count for player's moves as they are input.

F Flag. When set to 0, sub-routine 700 multiplies together all the C( ) values in a row to find the current line situation. With F set to 1, sub-routine 700 adds the value for that line

situation to the running totals for every position held in V( ).

G Line situation — found by multiplying all C( ) values for a line.

H Highest running total for a position.

IL Illegal move — an attempt to move into an already occupied position.

L( ) Line number situations. G is compared to these, and when a matching situation is found, the value of that line is given by the array value P( ).

M Move counter.

P( ) Points for a particular line situation.

V( ) Running totals for all positions in cube.

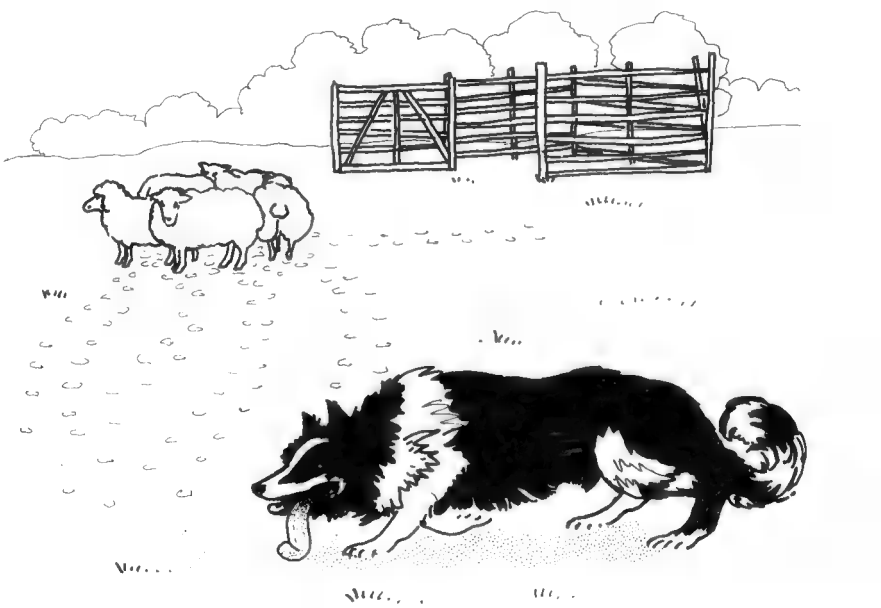
```
10 PRINTCHR$(6)CHR$(17):GOTO5000
20 INK4:PAPER3:PLOT11,16,"Type in your move"
25 PLOT13,18,"PLANE:":PLOT13,20,"ROW:":PLOT13,
22,"COLUMN:"
30 CO=1:K(1)=0:K(2)=0:K(3)=0:REPEAT:REPEAT:GET
A$
40 UNTILA$="1"ORA$="2"ORA$="3"ORA$="4"ORA$="D"
50 IF(A$<>"D")THENK(CO)=VAL(A$):PLOT21,16+2*CO
,A$:CO=CO+1
60 UNTILCO=4ORA$="D"
70 IFA$="D"THENGOSUB1800:GOTO20
80 IFC(K(1),K(2),K^3))>1THENIL=1:GOSUB1800:GOT
020
90 PING
100 A=K(1):B=K(2):C=K(3):C(A,B,C)=3:GOSUB400:G
OSUB300:GOSUB600:GOSUB1800
105 INK1:PAPER7
110 PLOT11,16,"My move is:"
120 PLOT13,18,"PLANE:":PLOT13,20,"ROW:":PLOT13
,22,"COLUMN:"
125 PLOT21,18,STR$(A):PLOT21,20,STR$(B):PLOT21
,22,STR$(C)
130 M=M+2:C(A,B,C)=2:GOSUB400:GOSUB300
140 IFM=64THENGOSUB2000:END
```

```
150 PING:GOSUB1800:GOTO20
200 PLOT1,2,"1 2 3 4   1 2 3 4   1 2 3 4   1 2
 3 4"
210 FORB=1TO4:FORA=1TO4:FORC=1TO4
220 GOSUB300
240 NEXT:NEXT:NEXT
250 PLOT8,24,"Press D to delete move":RETURN
300 X=(A-1)*10+2*C-1:Y=2*(B+1):P$=".":C1=C(A,B
,C):IFC1=2THENP$="O"
310 IFC1=3THENP$="X"
320 PLOTX,Y,P$:RETURN
400 FORV=1TO3:GOSUB700:NEXT
410 IFA<>BANDB<>CANDA<>CTHEN450
420 IFA=BTHENV=4:GOSUB700
430 IFA=CTHENV=5:GOSUB700
440 IFB=CTHENV=6:GOSUB700
450 IFA<>5-BANDA<>5-CANDB<>5-CTHEN530
460 IFA=5-BTHENV=7:GOSUB700
470 IFA=5-CTHENV=8:GOSUB700
480 IFB=5-CTHENV=9:GOSUB700
490 IFA=BANDA=5-CTHENV=10:GOSUB700
500 IFA=CANDA=5-BTHENV=11:GOSUB700
510 IFA=5-CANDB=CTHENV=12:GOSUB700
530 IFA=BANDB=CTHENV=13:GOSUB700
540 RETURN
600 H=0:FORA=1TO4:FORB=1TO4:FORC=1TO4
610 J=V(A,B,C):IFC(A,B,C)>1ORJ<HTHEN630
620 IFJ>HOR(J=H)ANDRND(1)>.5THENH=J:A1=A:B1=B:
C1=C
630 NEXT:NEXT:NEXT:A=A1:B=B1:C=C1:RETURN
700 FORT=1TO4:A1=A:B1=B:C1=C:ONVGOSUB850,860,8
70
705 IFV<4THEN720
710 A1=T:ONV-3GOSUB860,870,880,890,900,910,920
,930,940,950
720 IFF=1THENV(A1,B1,C1)=V(A1,B1,C1)+Z
730 IFF=0THENG=G*C(A1,B1,C1)
740 NEXT:IFF=0THENF=1:GOSUB1000:GOTO700
750 F=0:RETURN
850 A1=T:RETURN
860 B1=T:RETURN
870 C1=T:RETURN
880 A1=A:B1=T:C1=T:RETURN
890 B1=5-T:RETURN
```

```
900 C1=5-T:RETURN
910 A1=A:B1=T:C1=5-T:RETURN
920 B1=T:C1=5-T:RETURN
930 B1=5-T:C1=T:RETURN
940 B1=5-T:C1=5-T:RETURN
950 B1=T:C1=T:RETURN
1000 IFG=16THENGOSUB300:GOSUB1100:END
1010 IFG=81THENGOSUB300:GOSUB1200:END
1020 IF(G=6)ANDC(A,B,C)=2THENZ=-10:G=1:RETURN
1030 IFG/C(A,B,C)=6THENZ=0:G=1:RETURN
1040 FORN=1TO9:IFG=L(N)THENZ=P(N)
1050 NEXT:G=1:RETURN
1100 GOSUB1800:FORX=1TO30STEP7:FORY=16TO24STEP
2
1110 PLOTX,Y,"I WIN! ":NEXT:NEXT:EXPLODE:RETUR
N
1200 GOSUB1800:FORX=1TO30STEP8:FORY=16TO24STEP
2:PLOTX,Y,"YOU WIN!"
1210 NEXT:NEXT:EXPLODE:RETURN
1800 IFIL=1THENIL=0:PLOT13,16,"ALREADY OCCUPIE
D         ":SHOOT:WAIT(500)
1810 PLOT13,16,"                              ":PLOT
21,18,"    ":PLOT21,20,"      "
1820 PLOT21,22,"      "
1830 RETURN
1900 DIMC(4,4,4),V(4,4,4)
1910 G=1:FORA=1TO4:FORB=1TO4:FORC=1TO4
1920 V=0:IFA=BANDA=CTHENV=10
1930 IFA=CANDA=5-BTHENV=10
1940 IFA=5-CANDB=CTHENV=10
1950 IFA=BANDA=5-CTHENV=10
1960 V(A,B,C)=V:C(A,B,C)=1:NEXT:NEXT:NEXT
1970 FORN=1TO9:READL(N),P(N):NEXT:CLS:GOSUB200
:GOTO20
1980 DATA3,10,2,14,9,98,4,100,27,900,8,1000,6,
-14,18,-98,12,-100
5000 CLS:INK4:PAPER3:PRINTSPC(13);"3D Os and X
s":PRINTSPC(13);"============"
5010 PRINT:PRINT"In this game you must try to
get":PRINT"4 Os in a line to win."
5020 PRINT"The game is played on a 4x4x4 cube"
:PRINT"and each plane of the cube is"
5030 PRINT"shown on the screen. To type in":PR
INT"a move, you must type first the"
```

```
5040 PRINT"PLANE, then the ROW, and finally":P
RINT"the COLUMN where you want to"
5050 PRINT"place your O. If you change your":P
RINT"mind about a move, press D for"
5060 PRINT"DELETE. There will be a pause":PRIN
T"of a few seconds after your move"
5070 PRINT"before ORIC makes its reply."
5080 PRINT"Good luck - you'll need it!":PRINT"
Press C to continue."
5090 GETA$:IFA$<>"C"THEN5090
5100 GOTO1900
```

Sheepdog Trial



This program is based (not surprisingly) on a sheepdog trial. The sheep move at random until the dog gets within a certain distance: from then on the sheep always move directly away from the dog. You must use your dog to guide the sheep into the pen within a time limit.

If the sheep are allowed to wonder completely at random throughout the field, they can end up against a fence or jammed into a corner. As the sheep move away from any nearby dog, the player would have no choice but to retire his canine counterpart to some distance from the sheep, and hope that random movement will at some point bring the sheep out into a position where they can once more be herded.

The program therefore ensures that the sheep move no closer than one character position from the fence surrounding the field. It is then always possible for a dog to get 'behind' the sheep and herd them.

It is very difficult to herd more than two or three sheep into the pen. The captive sheep seem to possess an irresistable urge to wander back out of the pen just as your dog rounds up another!

The game is easily converted to run in real time by omitting the REPEAT from line 20 and changing line 100 to IFDD>8THEN140. Line 110 provides the shepherd's whistle as the dog is instructed to move in different directions. If this proves too irritating, just omit the line!

Program notes

10 Keyboard click and cursor off.

15-140 Dog move routine. The dog moves faster than the sheep: this is achieved by enabling the dog to move two or three times when the sheep only move once, and is the reason for the loop running from 15-140. The keyboard control routine runs from 20-100, 105 makes the dog face right or left according to its movement, 110 gives the shepherd's whistle, varying slightly for different directions of movement. 120 checks that the dog's new position is not occupied, and 130 moves the dog.

150-220 Sheep move routine. 150 finds the dog's x and y co-ordinates relative to BL, and 160 does similarly for the sheep. 165 finds the sheep's distance from the dog in terms of x and y displacement, and if the sheep are out of range they move randomly. Otherwise 170-180 find the appropriate movement which will take the sheep directly away from the dog. 185-190 find the random movement for sheep out of range, and 200 checks the new position for all sheep to make sure that the new position is unoccupied and not adjacent to a fence. 210 moves the sheep and 220 completes the loop.

230-270 Lines 230-250 check the pen to see if all the sheep are in yet. If not, and the time has not run out, the dog has another chance to move, 260. When the time runs out 270 checks if all the sheep have been rounded up.

280-440 Any dog rounding up all the sheep within the time limit is congratulated, 280. Other dogs are disqualified, 300-310. 350 saves the number of the best dog and its time. If all the dogs have taken part, 370, the winner is announced at 400. Otherwise the next dog is called to the field, 380. If no dog rounded up all the sheep, the game finishes with an abusive message, 410-420.

900-950 Initialisation routine.

960-1180 Game start routine. 1050-1090 draw the fence around the field and the pen fence. 1110-1150 randomly position the sheep within the field, 1115-1120 making sure they are away from the fence, and 1130-1140 ensuring they are not placed in the pen. 1150 pokes the sheep into their selected positions. 1160 chooses a random position for the dog, 1165 keeping it out of the pen, and 1170 ensuring the position isn't already occupied. 1180 pokes the dog to the screen.

2000-2020 User defined character routine.

5000 onwards — instructions, selection of number of sheep to be rounded up during the trials.

Important variables

BC Bottom corner inside pen.

BD Number of best dog in competition.

BT Time for best dog.

DC Dog count — gives dog number.

DD Dog direction of movement: used to select movement from array D( ).

DN Dog number — how many there are in the competition.

DP Dog position on-screen.

DR Dog rounding-up distance. Sheep further away than this move randomly.

DV Dog velocity. High values of DV mean the dog moves many times every time sheep move once.

DX Dog x coordinate relative to BL.

DY Dog y coordinate relative to BL.

LB Left bottom pen corner.

LT Left top pen corner.

ND New dog position.

PL Distance along left side of pen.

PT Distance along pen top.

SD Sheep direction of movement: used to select movement from D( ).

SN Sheep new position.

SP Sheep present position.

SR Number of sheep rounded up so far.

ST Total number of sheep to be rounded up.

SX Sheep x coordinate relative to BL.

SY Sheep y coordinate relative to BL.

TC Top corner inside pen.

X1 X coordinate difference between dog and sheep.

XN X movement when sheep is within dog rounding-up range.

Y1 Y coordinate difference between dog and sheep.

YM Y movement when sheep is within dog rounding-up range.

Symbols used

DS Dog symbol, either facing left or right.

FS Fence symbol.

PS Pen symbol.

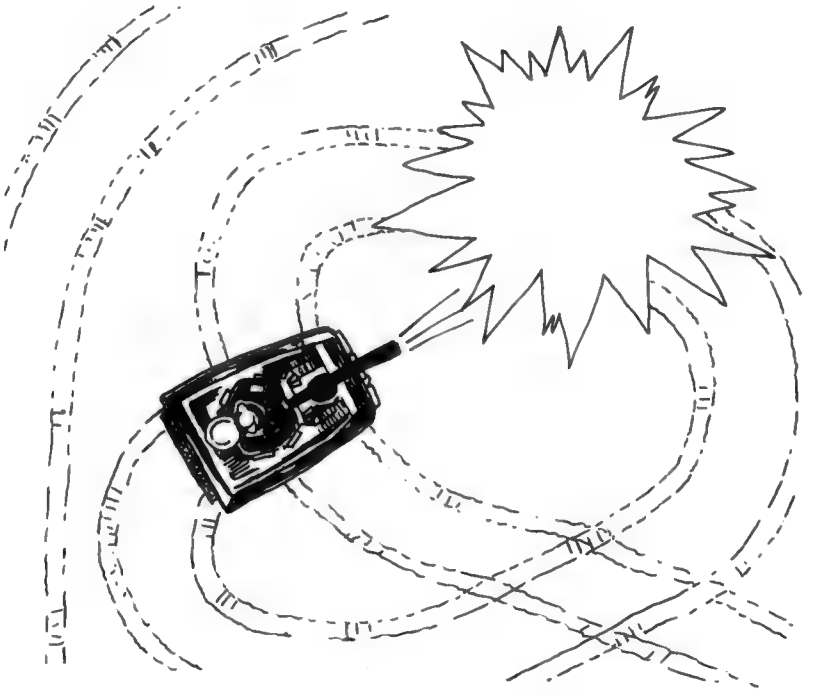
SS Sheep symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
15 FORD=1TODV
20 REPEAT:DD=8:GETA$:IFA$="W"THENDD=0
30 IFA$="E"THENDD=1
40 IFA$="D"THENDD=2
50 IFA$="C"THENDD=3
60 IFA$="X"THENDD=4
70 IFA$="Z"THENDD=5
80 IFA$="A"THENDD=6
90 IFA$="Q"THENDD=7
95 T=T+1:T$=STR$(T):PLOT23,25,T$
100 UNTILDD<8
105 DS=91:IFDD<4THENDS=95
110 FORG=1TO3:FORU=DDTO30:SOUND1,U,5:NEXT:NEXT
:SOUND1,0,0
120 ND=DP+D(DD):IFPEEK(ND)<>32THEN140
130 POKEDP,32:DP=ND:POKEDP,DS
140 NEXT
150 DY=INT((BL-DP)/LL)+1:DX=DP-BL+DY*LL
160 FORS=1TOST:SP=S(S):SY=INT((BL-SP)/LL)+1:SX
=SP-BL+SY*LL
165 Y1=SY-DY:X1=SX-DX:IFABS(Y1)>DRORABS(X1)>DR
THEN185
170 XM=0:IFX1<>0THENXM=X1/ABS(X1)
175 YM=0:IFY1<>0THENYM=Y1/ABS(Y1)
180 SN=SP+XM-YM*LL:SA=SN+XM-YM*LL:GOTO200
185 SD=INT(RND(1)*12):IFSD>7THEN220
190 SN=SP+D(SD):SA=SN+D(SD)
200 IFPEEK(SN)<>32ORPEEK(SA)=FSTHEN220
210 POKESP,32:S(S)=SN:POKESN,SS
220 NEXT
230 SR=0:B=LT+LL+1:C=B+PT-2
240 FORA=BTOC:IFPEEK(A)=SSTHENSR=SR+1
250 NEXT:B=B+LL:IFB<LBTHEN240
260 IF(T<TE)AND(SR<ST)THEN15
270 IFSR<STTHEN300
280 CLS:PRINTD$(DC);" rounded up ";ST;" sheep
in time ";T
290 GOTO350
300 PRINTD$(DC);" is disqualified."
310 PRINT"With his timing he'd be no good as a
":PRINT"watchdog."
350 IFT<BTTHENBT=T:BD=DC
370 WAIT(200):IFDC=DNTHEN400
```

```
380 PRINT:PRINT:PRINT"The next dog should now
proceed to":PRINT"the field.":GOTO1040
400 IFBT<TETHEN430
410 CLS:PRINT"These ";DN;" dogs were of such":
PRINT"poor quality that no rosettes"
420 PRINT"can be awarded.":END
430 PRINT"The winner was ";D$(BD);" in time ";
BT
440 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
905 LT=TL+374:RT=LT+5
910 DS=91:FS=92:PS=93:SS=94
915 FORZ=91TO95:GOSUB2000:NEXT
920 DC=0:DV=2:TE=200:BT=TE:DR=2:BD=0:D$(BD)="
    "
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:LB=LT+3*LL:PT=RT-LT:PL=L
B-LT
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1040
980 GOSUB5050
1040 T=0:DC=DC+1:PRINT"What is this dog's name
 ";:INPUTD$(DC)
1045 CLS:INK0:PAPER2
1050 FORA=TLTOTR:POKEA,FS:POKEA+DL,FS:NEXTA
1060 FORA=TLTOBLSTEPLL:POKEA,FS:POKEA+DT,FS:NE
XTA
1080 FORA=LTTORT:POKEA,PS:POKEA+PL,PS:NEXTA
1090 FORA=LTTOLBSTEPLL:POKEA,FS:POKEA+PT,PS:NE
XTA
1095 PLOT0,23,1:PLOT1,23,"BEST DOG:":PLOT11,23
,D$(BD)
1096 PLOT23,23,"BEST TIME:":B$=STR$(BT):PLOT34
,23,B$
1097 T=0:PLOT16,25,"TIME:"
1100 A=LT+INT(PT/2):POKEA,32
1105 TC=LT+LL+1:BC=LB-LL+1:DL=DL/LL
1110 FORS=1TOST
1115 C=0:SX=INT(RND(1)*(DT-3)):SY=INT(RND(1)*(
DL-3))
1120 SP=BL+SX-(SY*LL+2*(LL-1)):IFPEEK(SP)<>32T
HEN1115
1130 IFSP=TC+CORSP=BC+CTHEN1115
```

```
1140 C=C+1:IFC<PT-1THEN1130
1150 POKESP,SS:S(S)=SP:NEXTS
1160 DX=INT(RND(1)*DT+1):DY=INT(RND(1)*DL+1):D
P=BL+DX-DY*LL
1165 IF(DP>LT)AND(DP<LB)THEN1160
1170 IFPEEK(DP)<>32THEN1160
1180 POKEDP,DS:DL=DL*LL:GOTO15
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA0,0,16,49,17,31,17,17,18,18,63,18,18,
63,18,18,0,0,63,18,12,12,18,63
2020 DATA0,12,18,33,18,12,18,0,0,0,2,3,34,62,3
4,34
5000 CLS:PRINT"In Sheepdog Trial your dog must
":PRINT"round up sheep within the time"
5010 PRINT"limit. When your dog gets":PRINT"cl
ose to the sheep they will"
5020 PRINT"move away and can be herded into":P
RINT"the pen."
5030 PRINT:PRINT"Controls: use the keys around
 the S:"
5040 PRINTSPC(16);"Q W E":PRINTSPC(17);"A S D"
:PRINTSPC(18);"Z X C"
5050 PRINT:PRINT"How many dogs are taking part
 (1-9)";:INPUTDN
5055 IFDN<1ORDN>9THEN5050
5060 PRINT"Total number of sheep to be rounded
":PRINT"up, from 1 to 9";:INPUTST
5065 IFST<1ORST>9THEN5060
5070 TE=TE+ST*40:RETURN
```

Adder



In this game you control a tank which moves continuously around a rectangular playing area. Every time the tank hits one of the numbers dotted about the screen, that value is added to your score.

Sometimes in a game it is useful to have a shape whose direction of movement is clear from the way the shape looks. Common characters like this include arrows, spaceships, and, of course, tanks.

It is difficult to create 8 distinctive tank shapes on the Oric

because a 6 by 8 matrix is used, so in this game a tank is constructed from two individual symbols.

Each tank consists of a tank body, which is a 'block', and a tank gun. The gun symbols are stored in an array G( ). They just consist of single lines, with each line corresponding to an array value D( ). So if the tank is to move up the screen, its movement is given by D(0) and the tank gun G(0) is a vertical line which will be poked above the tank body. Similarly, D(3) indicates movement east, so G(3) is a horizontal line which will be poked east of the tank body. This is why the array values for G( ) in lines 910 and 915 repeat — the same line is used in two different positions to indicate a gun, the only difference being that it is poked onto a different side of the tank body in each case.

It is now easy to rotate the tank. To rotate clockwise the old gun is erased, 1 is added to the array values D( ) and G( ), and the new gun symbol G( ) is poked onto the tank body in the correct position by using D( ). Rotating anticlockwise is a similar process, except 1 is subtracted from the array values. In both cases checks must be made to ensure that the array value has not gone out of bounds.

Only three keys are needed to control the tank: one to move it, and the other two to rotate it clockwise or anticlockwise respectively. The resulting routines can be used in a variety of games, from tank battles to racing games where it is important to know which way you're going!

In the game that follows the tank moves automatically because this was found to be much faster than putting its movement under keyboard control. The player therefore can only control rotation of the tank, although it can be stopped by running it into a wall!

Program notes

10 Keyboard click and cursor off.

20-40 Number poke routine. 20 selects a random position on screen where the number will be poked, 30 checks the position is empty, and 40 pokes the number there.

50-80 Keyboard controls to check for tank rotation. 80 checks to see if there has been no direction change in which case the next section is omitted.

90-130 Tank rotate routine. 90 and 100 ensure that the array value has not gone out of bounds, 110 calculates the new gun position and checks that it is not a wall. 120 adds the number to the score if the new position contains a number. 130 erases the old gun and then draws a new one.

150-180 Tank move routine. 150 calculates the position two steps away from the tank body in the direction of movement, and checks that the position is not a wall. 160 adds the number to the score if the new gun position contains a number. 170 erases the old tank body and draws tank at its new position. 180 calls the subroutine which pokes numbers on screen if a number has just been erased by movement or rotation.

240-265 Score and time routine.

270-360 270 averages out your score to see if you qualify for a replay. Otherwise various comments are made on your success (or lack of it).

900-955 Initialisation routine.

965-1120 Game set-up routine. 1100 and 1105 draw walls around the screen edges, 1110 pokes all the numbers from 0 to 9 into random screen positions, and 1120 pokes the tank body plus gun into its start position at top left.

200-2020 User defined character creation routine.

5000 onwards — instructions.

Important variables

D Gives present direction of tank movement — used to select appropriate value from array D( ).

DL Distance along left of screen.

DT Distance along top of screen.

N Character code for number to be poked onto screen.

ND New direction of tank movement.

NG New gun position after rotation or movement.

PN Position of number on screen.

PP Present position of gun.

R Replay number — indicates how many replays have been made.

RN Replay number — the number of replays allowed before the game ends.

RS Replay score — this must be averaged on each game to get a replay.

S Score.

T Time.

TE Time end — if a replay score has not been achieved by this time the game ends.

Symbols used

NS Nine symbol — character code for number nine.

OS Nought symbol — character code for number zero.

TS Tank body symbol.

WS Wall symbol.

G(0) to G(7) Gun symbols. Each symbol is a single line, either vertical, horizontal, or diagonal.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 NX=INT(RND(1)*TD+1):NY=INT(RND(1)*SD+1)
30 PN=SP+NX-NY*LL:IFPEEK(PN)<>32THEN20
40 POKEPN,N:RETURN
50 N=0:ND=D:A$=KEY$:IFA$=CHR$(9)THENND=D+1
60 IFA$=CHR$(8)THENND=D-1
80 IFND=DTHEN150
90 IFND>7THENND=0
100 IFND<0THENND=7
110 NG=PP+D(ND):PG=PEEK(NG):IFPG=WSTHEN150
120 IFPG<>32THENS=S+PG-OS:N=PG
130 POKEPP+D(D),32:D=ND:POKEPP+D(D),G(D)
150 NP=PP+2*D(D):PG=PEEK(NP):IFPG=WSTHEN180
160 IFPG<>32THENS=S+PG-OS:N=PG
170 POKEPP,32:PP=PP+D(D):POKEPP,TS:POKENP,G(D)
```

```
180 IFN<>0THENGOSUB20
240 PLOTSC-7,24,"SCORE:":PLOTSC,24,STR$(S):T=T
+1:PLOTTI-6,24,"TIME:"
245 PLOTTI,24,STR$(T)
265 IFT<TETHEN50
270 R=R+1:IFS/R>=RSANDR<RNTHENT=0:GOTO50
280 CLS:ONRGOTO290,330,330,340
290 IFS>RS/4THEN300
295 PRINT"Back to your knitting granny!":GOTO3
50
300 IFS>RS/2THEN310
305 PRINT"You're no tank driver - a hearse ":P
RINT"suits you!":GOTO350
310 IFS>RS-10THEN320
315 PRINT"Keep trying - that's not too bad!":G
OTO350
320 PRINT"Tough luck - you almost got a replay
!":GOTO350
330 PRINT"Well done - that's a great score!":G
OTO350
340 PRINT"Fantastic! Perhaps it's even a recor
d!"
350 PRINT"Your score was ";S
360 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEEC:LL=40
905 SC=13:TI=25
910 NS=57:OS=48:TS=91:WS=92:G(0)=93:G(1)=94:G(
2)=95:G(3)=96
915 G(4)=93:G(5)=94:G(6)=95:G(7)=96
920 Z=TS:GOSUB2000:Z=WS:GOSUB2000:FORZ=93TO96:
GOSUB2000:NEXTZ
925 D=2:T=0:S=0:R=0:TE=400:RS=170:RN=4
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 BR=BR-LL:BL=BL-LL:DT=TR-TL:DL=BL-TL:TD=DT-
1:SD=(DL-LL)/LL
955 PP=TL+LL+1:SP=BL-LL-1
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK0:PAPER6
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
```

```
1110 FORN=OS+1TONS:GOSUB20:NEXTN:PLAY0,1,3,80
1120 POKEFF,TS:POKEFF+D(D),G(D):GOTO50
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA63,63,63,63,63,63,63,63,33,33,33,3
3,33,33,63,8,8,8,8,8,8,8
2020 DATA0,0,1,2,4,8,16,32,0,0,0,63,0,0,0,0,32
,16,8,4,2,1,0,0
5000 CLS:PRINT"In this game you must try to ad
d to"
5005 PRINT"your score by hitting numbers on":P
RINT"screen by running into them with"
5010 PRINT"your tank or hitting the numbers":P
RINT"as you swing the tank's gun."
5015 PRINT"If you get an average score of ";RS
:PRINT"in time ";TE;" you will be"
5020 PRINT"given a replay and timing will":PRI
NT"begin again from zero. A continued"
5025 PRINT"average of ";RS;" gives up to":PRIN
TRN;" replays."
5050 PRINT:PRINT"Controls are: left cursor key
 to":PRINT"rotate tank anticlockwise, and"
5060 PRINT"right cursor key to rotate it":PRIN
T"clockwise. Press any key to"
5070 PRINT"begin.":REPEAT:A$=KEY$:UNTILA$<>"":
RETURN
```

Minefield

In Minefield you have to find your way across a minefield riddled with potholes. You have a little help from your mine-detector, but its batteries are limited, so you must be decisive . . .

The mines on the field are randomly placed at the start, and their positions saved in the array M( ). Whenever a move is made, all nearby mine positions are examined, line 130, and checked to see if they correspond to positions directly surrounding the little man the player controls. The number of mines around the present spot is then displayed on-screen, although the player will have no idea in which direction the mines lie.

By comparing the number of mines surrounding one position to another adjacent spot, it is sometimes possible to deduce the mine locations. When this proves difficult, backtracking to a less dangerous path may be wisest, or you can always take the risk!

A trail of markers is left to show the safe path. Sufficiently skilled or reckless players may prefer to do without them, by changing the start of line 190 to POKEMP,32.

Program notes

10 Keyboard clock and cursor off.

20-100 Keyboard controls to move man in 8 compass directions. This loop repeats until a direction of movement is chosen.

110-190 Player move routine. 110 finds the new position, and checks if it is a pothole. Lines 130-180 check all the mine positions to see if any are adjacent to the present position, 140 counting all the mines around the new position. 170 checks in case the man has just trod on a mine, 190 pokes a safety marker to the previous position and draws the man at his new position.

200-240 200 checks if the top left has been reached, in which case the minefield has been crossed safely. 210 counts the number of times the detector has been used: it stops working after a certain period. 220 and 230 display the number of mines surrounding the present position.

260-310 Game end routine. It is always interesting to see how close to success you were, so after the explosion lines 260-280 reveal the locations of all the mines. 285 puts a marker at the last position, while 290 and 300 comment on the manner in which death came to pass.

900-955 Initialisation routine.

960-1210 Game set-up routine. 1100 and 1105 draw the edge of the minefield, while 1120-1140 randomly distribute potholes and mines about the playing area. 1170 clears away mines from the start and finish positions, otherwise it may be impossible to cross the minefield at all. 1190-1210 poke an exit at the top left of the minefield and place the man at his start position.

2000-2015 User defined character creation routine.

3000-3010 Routine when man falls down pothole.

5000 onwards — instructions plus setting of skill level.

Important variables

DC Detector count — how many times it has been used.

DE Detector end — detector gives out when DC reaches this value.

EP End position on minefield — just in from BR. Last position at which mines can be placed.

MA Mines about present position — their number.

MD My direction — used to select movement from array D( ).

MM Maximum number of mines that can be placed on the field.

MP My position.

NM Number of mines — count used as mines are placed at game-start.

NP New position after movement.

SP Start position on minefield — just in from TL. First position mines can be placed.

PF Pothole flag. Set to 1 if a fall takes place.

PM Probability of mine — used as mines are placed at game-start.

Symbols used

ES Explosion symbol.

HS Human symbol.

MS Mine symbol.

PS Pothole symbol.

SS Safety marker symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 REPEAT:MD=8:GETA$:IFA$="W"THENMD=0
30 IFA$="E"THENMD=1
```

```
40 IFA$="D"THENMD=2
50 IFA$="C"THENMD=3
60 IFA$="X"THENMD=4
70 IFA$="Z"THENMD=5
80 IFA$="A"THENMD=6
90 IFA$="Q"THENMD=7
100 UNTILMD<8
110 NP=MP+D(MD):IFPEEK(NP)=PSTHENPF=1:GOTO260
130 MA=0:FORA=1TONM:C=M(A):IFC<NP+D(7)ORC>NP+D
(3)THEN180
140 FORD=0TO7:IFC=NP+D(D)THENMA=MA+1
150 NEXTD
170 IFC=NPTHEN260
180 NEXTA
190 POKEMP,SS:MP=NP:POKEMP,HS
200 IFMP=TL+LL+1THENPLOT10,24,"    You escape!
   ":GOTO260
210 DC=DC+1:IFDC>DETHENMC=SC-1:S$=" DETECTOR F
AILS ":GOTO230
220 S$=STR$(MA)
230 PLOTMC,24,S$
240 GOTO20
260 FORA=1TONM:IFM(A)=-1ORM(A)=NPTHEN280
270 POKEM(A),MS
280 NEXTA
285 POKEMP,SS
290 IFPF=1THENGOSUB3000:END
300 PLOT6,24,"You stepped on a mine.":POKENP,E
S:EXPLODE
310 END
900 DIMM(20):TL=£BCA4:TR=£BCB0:BL=£BEB4:BR=£BE
90:LL=40
910 MS=42:HS=91:PS=92:SS=93:ES=94
915 Z=HS:GOSUB2000:Z=PS:GOSUB2000:Z=SS:GOSUB20
00:Z=ES:GOSUB2000
920 DC=0:MA=0:NM=0:PF=0:MM=10:PP=.07:P^=.2
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:SP=TL+LL+1:EP=BR-LL-1:SC
=10
955 MC=SC+13:B=SP:MP=EP
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
```

```
1000 CLS:INK2:PAPER0
1100 FORA=TLTOTR:POKEA,PS:POKEA+DL,PS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,PS:POKEA+DT,PS:NE
XT
1120 FORA=BTOB+DT-2:R=RND(1):S=32:IFR<PPTHENS=
PS:GOTO1140
1125 IFNM=MMTHEN1140
1130 IFR<PMTHENNM=NM+1:M(NM)=A
1140 POKEA,S:NEXT:B=B+LL:IFB<=EPTHEN1120
1170 FORA=1TONM:C=M(A):FORZ=0TO7:IFC=MP+D(Z)OR
C=SP+D(Z)THENM(A)=-1
1180 NEXT:NEXT
1190 PLOTSC,24,"MINES PRESENT"
1200 POKETL,32:POKESP,32:POKEMP,HS
1210 GOTO220
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA14,14,4,31,4,4,10,17,12,18,33,33,18,1
2,0,0,0,48,40,36,40,48,32,32
2015 DATA2,32,9,37,8,3,33
2016 DATA7
3000 PLOT5,24,"You fell into a pothole"
3010 FORQ=1TO100:SOUND1,Q,4:NEXT:PLAY0,0,0,0:R
ETURN
5000 CLS:PRINT"You are trapped in a minefield"
:PRINT"full of potholes. Try to"
5005 PRINT"escape through the exit at the":PRI
NT"top left. Your mine detector tells"
5010 PRINT"you how many mines are directly"
5015 PRINT"around you. But its batteries":PRIN
T"can run out - so be quick! When you"
5020 PRINT"move, safety markers are left":PRIN
T"behind."
5030 PRINT:PRINT"Controls - to move, press the
":PRINT"keys surrounding the S key:"
5040 PRINT:PRINTSPC(16);"Q W E":PRINTSPC(17);"
A S D":PRINTSPC(18);"Z X C"
5045 PRINT"To move north, for example, press":
PRINT"W."
5050 PRINT:PRINT"Input your skill level, 1 to
10,":PRINT"1 gives few mines, 10, many";
5060 INPUTSL
5070 IFSL<1ORSL>10THEN5050
5080 MM=MM+SL:DE=DT+(DL/LL)+SL:RETURN
```

Alien Attack

You are fighting to save earth from the alien hordes that fall from the skies. The monstrous invaders cannot survive the touch of earthly skin, and shatter into a thousand fragments if you touch them. But any aliens that fall to the ground encyst themselves in a toughened shell, and once that pile of shells reaches over your head, you will DIE!

This is a relatively brief program, but the game is great fun to play. The aliens are poked randomly to the screen using subroutine 10, and the number of aliens that fall at any one time can be varied just by changing the value of BN, line 5070. For a really

tough game, try playing with 3 aliens dropping down at once — you have to be very quick to get them all!

Program notes

5 Keyboard click and cursor off.

10-20 Alien placement routine. 10 picks a random position along the top of the screen, 20 saves that position in the array B( ) so the alien can be moved later.

30-80 Alien drop routine. 30 finds the alien's current position and checks that the alien is still there: it might have been destroyed in the previous move. 40 finds the alien's new position and erases it from its old position, while 60 turns it into a shell body symbol if it is about to strike the ground or another alien corpse. 70 checks if the alien has hit the man, and 75 moves it into its new position if that position is empty. 80 completes the loop which moves all the aliens.

90-120 Keyboard control routine. 90-100 check if the man is being moved left, right, or kept stationary. 110 finds the man's new position, and 120 draws him at that position.

125-140 Game end routine. The score is based purely on the length of time for which the man survives.

900-920 Initialisation routine.

960-1100 Game set-up routine. 1050 ensures that the foreground colour at the lower part of the screen is red, so all the alien cysts are suitably gruesome.

2000-2020 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

BP Body position for alien — where it is in screen memory.

ML Man left — furthest left man may move on-screen.

MP Man position.

MR Man right — furthest right man may move on-screen.

NB New body position for alien.

NP New position for man.

PB Value found at new position of alien body.

Symbols used

BS Body symbol for alien.

ES Explosion symbol when alien strikes man.

MS Man symbol.

WS Worse body symbol — used when alien becomes a cysted corpse.

```
5 PRINTCHR$(6)CHR$(17):GOTO900
10 P=TL+INT(RND(1)*(TR-TL)+1):IFPEEK(P)=BSTHEN
10
20 B(A)=P:RETURN
30 FORA=1TOBN:BP=B(A):IFPEEK(BP)<>BSTHENGOSUB1
0:BP=B(A)
40 NB=BP+LL:PB=PEEK(NB):POKEBP,32:IFPB=WSANDBP
<MRTHENF=1
60 IFNB>BRORPB=WSTHENPOKEBP,WS:PB=WS
70 IFPB=MSTHENPING
75 IFPB=32THENB(A)=NB:POKENB,BS
80 NEXT:RETURN
90 REPEAT:A$=KEY$:IFA$=CHR$(8)THENN=-1
95 IFA$=" "THENN=0
100 IFA$=CHR$(9)THENN=1
110 NP=MP+N:POKEMP,32:IFNP<MLORNP>MRTHENNP=MP
120 MP=NP:POKEMP,MS:GOSUB30:T=T+1:PLOT17,24,ST
R$(T):UNTILF=1
125 DORX=1TO5:PING:NEXT
130 CLS:PRINT"You lasted for time ";T:PRINT"Th
is was at skill level ";SL
140 END
900 TL=£BBD7:TR=£BBF0:BL=£BED1:BR=£BEF1:LL=40
910 MS=93:BS=94:WS=95:ES=96
915 Z=MS:GOSUB2000:Z=BS:GOSUB2000:Z=WS:GOSUB20
00:Z=ES:GOSUB2000
920 T=0:F=0:ML=TL+16*LL:MR=TR+16*LL:MP=ML+16
960 CLS:PRINT"Do you want instructions (Y/N)";
```

```
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK4:PAPER3:PLOT11,24,"TIME:"
1050 FORZ=ML+39TOBLSTEPLL:POKEZ,1:NEXT
1100 GOSUB10:POKEMP,MS:GOTO90
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA14,14,4,31,4,4,10,17,0,51,18,18,63,45
,63,63
2020 DATA63,33,33,33,33,33,33,63,8,2,32,1,16,4
,0,0
5000 CLS:PRINT"You must try to stop the aliens
":PRINT"from falling to the bottom"
5005 PRINT"of the screen by popping them by":P
RINT"placing your man underneath"
5010 PRINT"them. If you don't stop the aliens"
:PRINT"their corpses will form a pile"
5015 PRINT"which will build up until it":PRINT
"reaches you and buries you, then"
5020 PRINT"the game ends.":PRINT:PRINT"Use the
 cursor keys to move the man"
5030 PRINT"left and right."
5040 PRINT:PRINT"Use the space bar to stop."
5050 PRINT:PRINT"Input your skill level, from
1 to 5,":PRINT"1, easy, to 5, difficult";
5060 INPUTSL:IFSL<1ORSL>5THEN5050
5070 BN=SL:RETURN
```

Grave Robber

Here, you must try and collect as much gold as possible while avoiding the poison arrows which fall from the ceiling above. As if this wasn't enough, you must be very quick before the descending ceiling crushes you to death!

To make this game a little easier, protective bases are placed across the screen so that the man can seek shelter from the shoals of falling arrows. The horizontal and vertical dimensions of the walls and their number are determined in line 925. More confident players can eliminate the protective walls completely by adding 1101 GOTO1150.

Program notes

10 Keyboard click and cursor off.

15-25 Arrow explosion routine. When an arrow hits the man or the floor, 15, it explodes, 20, and a new arrow is created to replace it, 25.

30-60 Move arrow routine. 40 checks if the arrow has hit anything or reached floor level, 50 pokes the arrow to its new position.

70-80 Keyboard control routine.

90-140 Player move routine. 90 checks that the present position is occupied: if it is empty the man has been destroyed by an arrow and the game is over. 100 ensures the man does not move off-screen left or right. When the side of the screen is reached, a gold bar is removed from the pile, and the gold bar flag GB is set to 1, 110. When the bar is deposited on the left side of the screen, GB is rest to and a gold bar is added to the pile on that side, 120. 130 checks if all the bars have been collected, if not 135 exmines the new position of the man to check if there is an arrow there. 140 moves the man to the new position.

150-160 If the maximum number of arrows is not yet falling, 150 may add a new arrow, while 160 loops back to begin the whole sequence again.

170-240 Game end routine. Various messages are given depending on the success or failure of your robbery.

900-960 Initialisation routine.

961-1190 Game set-up routine. 1100 draws the floor, 1105-1140 place the protective walls above the floor: these are gradually destroyed as arrows hit them during the game. 1150 pokes the pile of gold bars to the right of the screen. 1190 calls the routine that lowers the ceiling, and the game begins.

1200-1220 Lower ceiling routine. This is called every time a gold bar is removed from the pile.

2000-2020 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

AN New arrow position.

DW Distance between protective walls.

GB Gold bar flag — set to 1 when gold bar is taken.

GC Gold bar count.

GN Number of bars collected by player.

GP Gold bar position on right.

GV Value of bar in pounds.

G1 Top left gold bar position.

HW Horizontal dimension of protective walls.

MA Maximum number of arrows to be dropped at one time.

ML Man's leftmost allowable position.

MM Man's movement.

MP Man's position.

MR Man's rightmost allowable position.

NA Number of arrows presently dropping.

NG Number of gold bars altogether.

NP New position for man.

NW Number of protective walls.

PA Position of arrow.

SL Skill level.

VW Vertical dimension of protective walls.

WP Wall position.

WU How far walls are up from floor.

Symbols used

AS Arrow symbol.

ES Explosion symbol.

FS Floor symbol.

GS Gold bar symbol.

MS Man symbol.

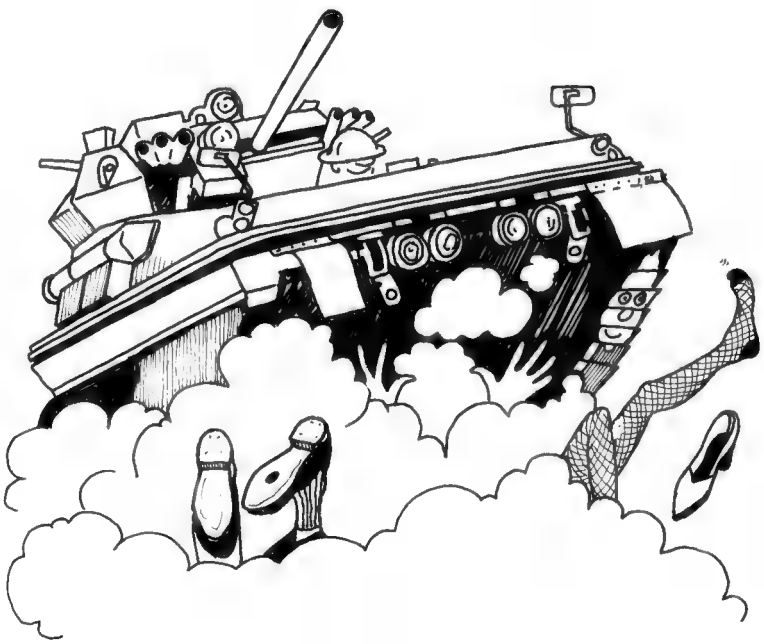
WS Wall symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
15 S=32:IFAP=FSTHENS=FS
20 POKEAN,ES:WAIT(5):POKEAN,S
25 P(A)=TL+INT(RND(1)*RA+2):RETURN
30 FORA=1TONA:PA=P(A):AN=PA+LL:AP=PEEK(AN):POK
EPA,32
40 IFAP<>32ORAN>BLTHENPING:GOSUB15:GOTO60
50 POKEAN,AS:P(A)=AN
60 NEXT
70 A$=KEY$:IFA$=CHR$(9)THENMM=1
75 IFA$=" "THENMM=0
80 IFA$=CHR$(8)THENMM=-1
85 T=T+1:T$=STR$(T)+"    ":PLOT22,24,T$:IFT=TET
HENT=0:GOSUB1200
90 PM=PEEK(MP):IFPM=32THEN170
100 NP=MP+MM:IFNP<MLORNP>MRORNP=MPTHEN150
110 IFNP=MRANDGB=0THENGB=1:POKEGP,32:GP=GP+LL:
GOSUB1200
120 IFNP=MLANDGB=1THENGB=0:POKEG1,GS:G1=G1-LL:
GN=GN+1
130 IFGN=NGTHEN200
135 IFPEEK(NP)=ASORTR>MPTHENEXPLODE:WAIT(500):
GOTO170
140 POKEMP,32:MP=NP:POKEMP,MS
150 IFRND(1)<SLANDNA<MATHENNA=NA+1:P(NA)=TL+IN
T(RND(1)*RA+2)
160 GOTO30
170 CLS:PRINT:PRINT"You perished from the pois
on at the":PRINT"tip of the arrow, but your"
180 PRINT"grateful relatives ";
190 IFGN>0THENPRINT"got ";GN*GV;" thousand pou
nds for the gold.":GOTO240
195 IFGN=0THENPRINT"sold your clothes  for a f
ew pounds.":GOTO240
200 CLS:PRINT:PRINT"You return home having col
lected"
```

```
205 PRINTGN*GV;" thousand pounds worth of gold
."
240 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 AS=91:ES=92:FS=93:GS=94:MS=95:WS=96
915 FORZ=91TO96:GOSUB2000:NEXT
920 NG=6:GN=0:NA=4:MA=NA:GC=0:GB=0:WU=3:T=0:TE
=50
925 HW=3:VW=2:NW=3
950 DT=TR-TL:RA=DT-2:DW=INT((DT-NW*HW)/(NW+1))
955 WP=BL+DW-WU*LL
960 ML=BL-LL:G1=ML-LL:MP=ML:MR=BR-LL:GP=BR
961 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK1:PAPER7
1100 POKEBL-1,0:FORA=BLTOBR:POKEA,FS:NEXT:POKE
MP,MS
1105 FORA=1TONW:FORH=0TOHW-1:FORV=0TOVW-1
1110 P=WP+H-V*LL:POKEP,WS:NEXTV:NEXTH
1140 WP=WP+HW+DW:NEXT
1150 GC=GC+1:IFGC<=NGTHENGP=GP-LL:POKEGP,GS:GO
TO1150
1160 PLOT15,24,"TIME:"
1190 GOSUB1200:GOTO30
1200 FORA=TL-LLTOTR-LL:POKEA,32:NEXT
1205 TL=TL+LL:TR=TR+LL
1210 FORA=TL-LLTOTR-LL:POKEA,FS:NEXT
1220 RETURN
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA8,8,8,8,8,42,28,8,8,2,32,1,16,4,0,0,6
3,63,63,63,63,63,63,63
2020 DATA63,33,33,33,33,33,33,63,14,14,4,31,4,
4,10,17,21,42,21,42,21,42,21,42
5000 CLS:PRINT"As a grave robber you risk deat
h":PRINT"to snatch gold bars from"
5005 PRINT"ancient tombs. There are many traps
":PRINT"in the tombs, and only the "
5010 PRINT"most skilled robbers can evade the"
:PRINT"poison arrows which fall from"
5020 PRINT"above. The gold bars are so heavy":
PRINT"that you can only carry one at"
5030 PRINT"a time. You must drop the bar near"
```

```
:PRINT"the exit, which is on the left,"
5035 PRINT"before returning to the right for":
PRINT"more bars."
5040 PRINT"Watch out for the descending ceilin
g!"
5045 PRINT:PRINT"Controls - use the cursor key
s to":PRINT"move right or left."
5050 PRINT:PRINT"How skilled a robber are you,
 1,":PRINT"hopeless, to 5, master crook";
5060 INPUTSL:IFSL<1ORSL>5THEN5050
5070 TE=TE-5*SL:MA=MA+SL:GV=SL*2:SL=SL*.02:RET
URN
```

Manhunt



You are a madman who has seized a poorly-guarded tank and gone on the rampage. With no thought for others you mow your victims down under the huge treads of your machine. How many more will perish before you are caught?

Although Manhunt is a completely different program to Adder, it once again makes use of the tank as the player's symbol. By careful program writing, part of the Adder program can be used as the basis for a new game.

In Manhunt the aim of the game is to run down as many randomly moving victims as possible in a given time. The game is made more difficult by the fact that any man hit by the tank is

buried on the screen beneath a cross, through which the tank cannot move. Manoeuvring thus becomes harder as the game progresses. The number of men is kept at a reasonable level by replacing the 'dead' men elsewhere on the screen.

To save programming time, it is possible to load Adder, delete lines 300-360, and then type in just the following lines which differ in the listings: 40, 50, 110, 120, 150, 160, 180-220, 265-290, 905-925, 1000, 1110, 2010 onwards.

As the programs have a number of lines in common, the program notes describe only those lines which differ, and the reader is referred to Adder for a full description of any other sections.

Program notes

40 This pokes a man to a random position on-screen, saving that position in the array M( ).

50 Keyboard control routine.

110 Checks tank's new position. The tank will not move if it hits a wall or a cross.

120 The only other object the tank can hit must be a man, and this line pokes a cross to the man's position.

150 Gun rotate routine. If the swinging gun hits a man he is replaced by a cross.

160 If the gun hits anything else, it can't move.

180-220 Man move routine. 180 checks if there is now a cross at the man's position, in which case he must be dead, and should be placed elsewhere on-screen by calling sub-routine 20. 185 ensures that the man moves less frequently for less skilled players, and 190 calculates a random movement for him. 200 checks to see if the man has blundered into the tank or gun during his random cavortings, otherwise 210 erases him from the old position and draws him at the new one. 220 completes the loop to move the remaining men.

265-290 Game end routine. Gives the score.

1110 Places the men randomly on-screen at the start of the game.

2000-2030 User defined character routine.

5000 onwards — instructions and setting of skill level.

Important variables

The only new variables introduced in Manhunt are:

MN Number of men to move around.

MT Total number of men to be killed in time limit TE.

Symbols used

The only symbols which differ from those in Adder are:

CS Cross symbol.

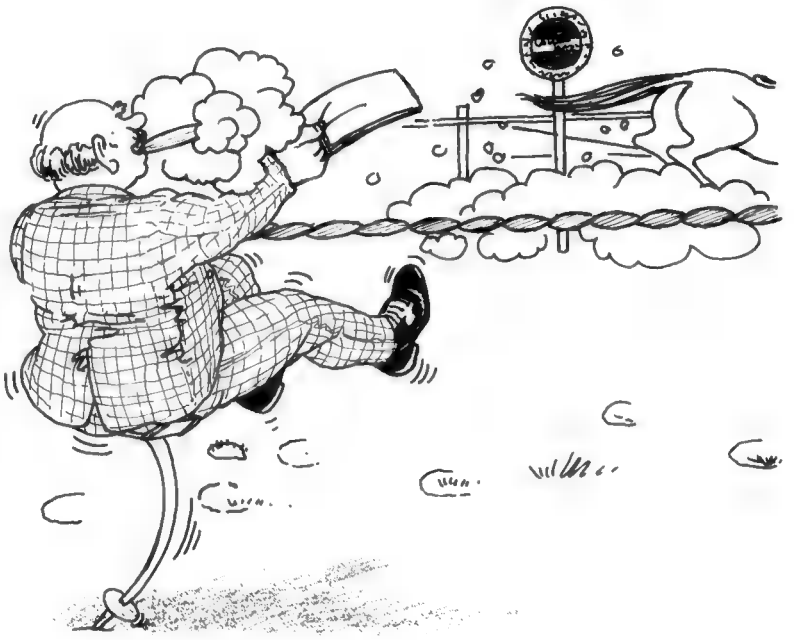
N Man symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 NX=INT(RND(1)*TD+1):NY=INT(RND(1)*SD+1)
30 PN=SP+NX-NY*LL:IFPEEK(PN)<>32THEN20
40 M(A)=PN:POKEPN,N:RETURN
50 ND=D:A$=KEY$:IFA$=CHR$(9)THENND=D+1
60 IFA$=CHR$(8)THENND=D-1
80 IFND=DTHEN150
90 IFND>7THENND=0
100 IFND<0THENND=7
110 NG=PP+D(ND):PG=PEEK(NG):IFPG=WSORPG=CSTHEN
150
120 IFPG<>32THENPOKENG,CS:GOTO150
130 POKEPP+D(D),32:D=ND:POKEPP+D(D),G(D)
150 NP=PP+2*D(D):IFPEEK(NP)=NTHENPOKENP,CS:GOT
O180
160 IFPEEK(NP)<>32THEN180
170 POKEPP,32:PP=PP+D(D):POKEPP,TS:POKENP,G(D)
180 FORA=1TOMN:IFPEEK(M(A))=CSTHENS=S+1:GOSUB2
0
185 PN=M(A):IFRND(1)>SLTHEN220
190 M=INT(RND(1)*8):NM=PN+D(M):P=PEEK(NM)
200 IFP=TSORP=G(D)THENPOKEPN,CS:GOTO220
210 IFP=32THENPOKEPN,32:M(A)=NM:POKENM,N
220 NEXTA
```

```
240 PLOTSC-7,24,"SCORE:":PLOTSC,24,STR$(S):T=T
+1:PLOTTI-6,24,"TIME:"
245 PLOTTI,24,STR$(T)
265 IFS<MT ANDT<TETHENS0
270 CLS:PRINT"The mad driver killed ";S;" men"
:PRINT"before his petrol ran out. This"
280 PRINT"was at skill level ";SL*10:PRINT"A t
otal score of ";TE-T+SL*S*10
290 PLAY0,0,0,0:END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEEC:LL=40
905 SC=15:T1=25
910 CS=43:TS=91:WS=92:G(0)=93:G(1)=94:G(2)=95:
G(3)=96:N=38
915 G(4)=93:G(5)=94:G(6)=95:G(7)=96
920 FORZ=91TO96:GOSUB2000:NEXT:Z=N:GOSUB2000
925 D=2:T=0:S=0:TE=400:MN=5:MT=15
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 BR=BR-LL:BL=BL-LL:DT=TR-TL:DL=BL-TL:TD=DT-
1:SD=(DL-LL)/LL
955 PP=TL+LL+1:SP=BL-LL-1
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK1:PAPER3
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
1110 FORA=1TOMN:GOSUB20:NEXTA
1120 POKEPP,TS:POKEPP+D(D),G(D):PLAY0,1,3,80:G
OTO50
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA63,63,63,63,63,63,63,63,33,33,33,3
3,33,33,63,8,8,8,8,8,8,8,8
2020 DATA0,0,1,2,4,8,16,32,0,0,0,63,0,0,0,0,32
,16,8,4,2,1,0,0
2030 DATA14,14,4,31,4,4,10,17
5000 CLS:PRINT"In Man Hunt you are a lunatic w
ho":PRINT"has managed to seize a tank."
5005 PRINT"Fortunately, it has no ammunition,"
:PRINT"but you can still cause havoc"
5010 PRINT"by running over any passers-by":PRI
NT"or hitting them with your gun."
```

```
5015 PRINT"Dead men will be buried beneath":PR
INT"crosses - you cannot drive over"
5020 PRINT"these. Your petrol will run out":PR
INT"in time ";TE;" although if you"
5025 PRINT"kill ";MT;" men your treads will":P
RINT"probably be unable to move due to"
5030 PRINT"the bones jamming them."
5050 PRINT:PRINT"Controls are: left cursor key
 to":PRINT"rotate tank anticlockwise, and"
5060 PRINT"right cursor key to rotate it":PRIN
T"clockwise. Please input"
5070 PRINT"your skill level, 1, easy, to 10, h
ard";:INPUTSL
5080 IFSL<1ORSL>10THEN5050
5090 SL=SL/10:RETURN
```

Derby Day



The shrewd gambler can clean up on this one: with a careful look at the form, it's not too hard to select a likely looking winner for the next race. Then it is sit back and cheer your choice on as the horse sprints towards the finishing line!

Every horse which takes part in the races has a movement number associated with it. This is generated by the computer equivalent of throwing two dice: two random numbers from 1 to 6 are found and added together. This value is stored in the array H( ).

When a race is run, a second random number is generated in exactly the same way. If this matches the movement number in

the array H( ), the horse is moved forward. Obviously horses with certain movement numbers are at a severe disadvantage. A number like 12 is not going to come up very often! Their poor chances are reflected in the odds given at the beginning of each race. The odds are thus a good guide to the form of the horses, but a random element is always included so that the odds are not a direct reflection of a horse's chances. An interesting idea to extend the program would be to run a number of races internally and produce these as a form guide. Some horses might be penalised according to ground conditions, and so on. There is plenty of scope here for experimentation.

Program notes

10 Keyboard click and cursor off

15-90 Horse movement routine. 15 generates a random number, and 20 compares this to the horse movement number. If the two do not match, the horse does not move. Any horse moving has its new position checked in 30, to see if it has reached the finishing line. If so, the flag F is set to 1 to indicate a winner, and line 80 then ensures that the loop terminates with this horse. There are thus no dead-heats, as movement ends as soon as a horse crosses the line. 40 calculates all the new positions for parts of the horse's body, based on the position of its front leg, and 50 erases the horse. 60 finds the new positions, and the last part of the line ensures that the legs are straight and at an angle on successive moves. 70 draws the horse at its new position, 80 ends the loop early if the horse has just won, otherwise 90 continues the race.

100-170 Winnings and losses routine. 105 checks if you picked the winner. If you did, 110 calculates your winnings and 115 displays them. If you are broke, 120 ends the game. Otherwise, 130 checks if you have reached the winnings target, and you are given the option of continuing or retiring gracefully, 140. 150 sets up a new race, 160 and 165 giving various messages depending on whether you gave up betting or ran out of money.

900-920 Initialisation routine.

960-1070 Game set-up routine. 1005 adds one to the race number and picks a random number of horses to run in the next race. 1010

picks a random name at which to start reading the names of the horses to run in the race. 1015 generates horse movement numbers and 1020 saves that number and the horse's name to the arrays H( ) and H$( ) respectively. 1040-1070 find the odds for each horse, the very slow ones having their odds increased by 1050! 1060 adds or subtracts a random amount from the odds, the latter only when the horse has a very good chance of winning.

1080-1190 Start race routine. 1090-1150 print out information about the race, and takes and checks your bet. 1160 picks each horse position, going up the screen with 4 lines between each horse — this can be amended to bring them closer together or to spread them out. 1170-1180 calculates the positions for different parts of the horse's body, and pokes it on-screen. 1190 pokes the finishing line into place.

2000-2030 User defined character creation routine.

3000-3010 Horse name data.

Important variables

DV Dice value — used to pick movement number for each horse.

F Flag — set to 1 when a horse crosses the finish line.

HB Horse back leg position.

HF Horse front leg position.

HH Horse head position.

HN Horse new front leg position after movement.

HT Horse torso position.

LH Largest number of horses in race.

NH Number of horses in race.

OD Odds for horse.

PH Screen position of horse's front leg at start of race.

R Race number.

SH Smallest number of horses in race.

SP Start position for poking horses on-screen at beginning of race.

T1, T2 Tab positions to align horse names and odds on-screen.

TH Total number of horse names available.

TM Target money to be won by player.

WH Winning horse number.

YB Your bet.

YH Your chosen horse.

YM Your present money.

YW Your winnings.

Symbols used

B1 Set equal to BD or BS, the back leg symbols.

BD Back leg diagonal.

BS Back leg straight.

F1 Set equal to FD or FS, the front leg symbols.

FD Front leg diagonal.

FL Finish line symbol.

FS Front leg straight.

HS Head symbol.

OS Nought symbol — symbol for number zero.

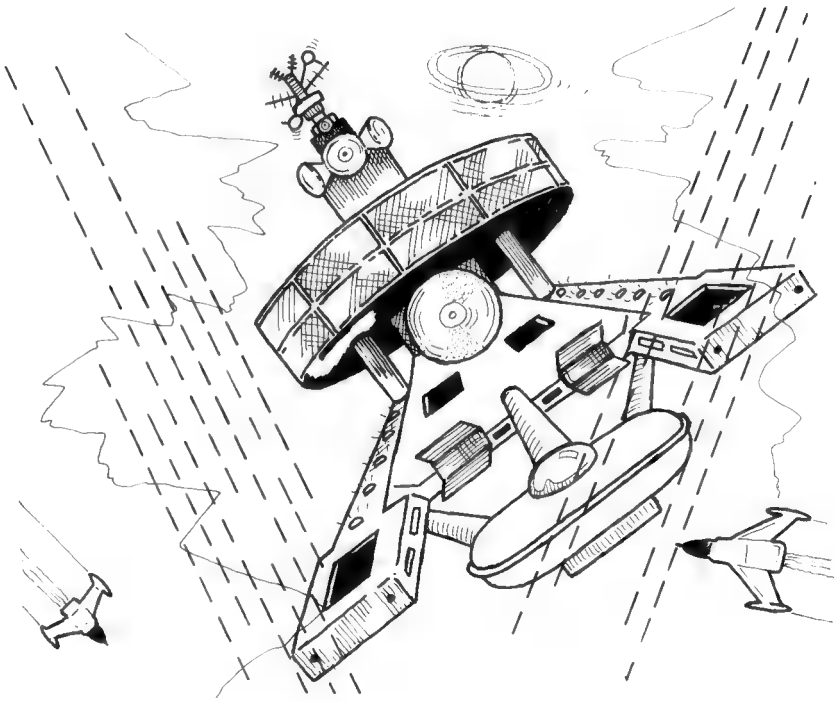
TS Torso symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
15 FORA=1TONH:H1=INT(RND(1)*6+1):H2=INT(RND(1)
*6+1)
20 IFH(A)<>H1+H2THEN90
30 HF=P(A):PF=PEEK(HF):HN=HF+1:IFPEEK(HN)=FLTH
ENF=1
40 HB=HF-2:HT=HF-LL-1:HH=HF-LL*2
```

```
50 POKEHB,32:POKEHT,32:POKEHF,32:POKEHH,32
60 HB=HB+1:HT=HT+1:HH=HH+1:B1=BS:F1=FS:IFPF=FS
THENB1=BD:F1=FD
70 POKEHB,B1:POKEHT,TS:POKEHN,F1:POKEHH,HS:P(A
)=HN
80 IFF=1THENWH=A:A=NH
90 NEXTA:IFF=0THEN15
100 PRINT:PRINT:PRINT"The winner was horse num
ber ";WH;H$(WH)
105 IFYH<>WHTHEN120
110 YW=YB*O(WH)
115 PRINT"You get £";YW;".":YM=YM+YW:GOTO130
120 YM=YM-YB:IFYM=0THEN165
130 IFYM>=TMTHEN150
135 PRINT"You now have £";YM;"."
140 PRINT"Do you wish to continue betting";:IN
PUTA$
150 IFA$="Y"THENF=0:GOTO1000
160 PRINT"You have finished with £";YM;"."
170 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 HS=91:TS=92:BS=93:FS=94:BD=123:FD=124:FL=1
25:OS=48
915 FORZ=91TO94:GOSUB2000:NEXT:FORZ=123TO125:G
OSUB2000:NEXT
920 YM=200:TM=2000:SH=2:LH=5:TH=8:R=0:T1=3:T2=
20
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000
1000 CLS:INK0:PAPER2
1005 R=R+1:NH=INT(RND(1)*(LH-SH)+SH)
1010 NA=INT(RND(1)*(TH-LH)):FORB=1TONA:READA$:
NEXT
1015 FORA=1TONH:H1=INT(RND(1)*6+1):H2=INT(RND(
1)*6+1)
1020 H(A)=H1+H2:READA$:H$(A)=A$
1040 DV=H(A):IFDV>7THENDV=14-DV
1050 OD=10-DV:IFDV<4THENOD=OD*2
1060 EX=INT(RND(1)*OD):IFRND(1)>0.5ANDDV>4THEN
EX=-EX
1070 O(A)=OD+EX:NEXTA:RESTORE
1080 CLS
1090 PRINT"Race number ";R:PRINTNH;" horses ru
n."
```

```
1100 PRINTTAB(T1);"Name";TAB(T2);"Odds"
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
1110 FORA=1TONH:PRINTA;H$(A);TAB(T2);O(A);"to
1":NEXTA
1120 PRINT"What number horse will you back";:I
NPUTYH
1130 YH=INT(YH):IFYH<1ORYH>NHTHEN1120
1140 PRINT"How much will you bet";:INPUTYB
1150 YB=INT(YB):IFYM-YB<0ORYH>NHTHEN1120
1160 CLS:SP=BL-LL+3:FORA=1TONH:PH=SP-(A-1)*4*L
L:P(A)=PH
1170 HF=PH:HB=HF-2:HT=HF-LL-1:HH=HF-(LL*2):N=H
B-1
1180 POKEHB,BS:POKEHT,TS:POKEHF,FS:POKEHH,HS:P
OKEN,OS+A
1190 NEXTA:FORB=TRTOBRSTEPLL:POKEB,FL:NEXTB:GO
TO15
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA0,0,0,0,32,32,56,56,1,1,1,1,63,63,63,
63,1,1,1,1,1,1,0,0
2020 DATA32,32,32,32,32,32,0,0,1,2,4,8,16,0,0,
0,32,16,8,4,2,0,0,0
2030 DATA12,12,12,0,0,12,12,12
3000 DATA" RED FRED "," BULLET "," STAR "," KA
NGAROO "
3010 DATA" SMOKEY "," GUTSY "," THE FLIER ","
SLOWCOACH "
5000 CLS:PRINT"You have £";YM;"to bet. You":PR
INT"must aim to end up with"
5010 PRINT"£";TM;" or more. The odds given":PR
INT"for each horse are a guide to its"
5020 PRINT"chances, but outsiders can win!"
5030 PRINT:PRINT"Press any key to continue.":G
ETX$:RETURN
```

Starship Warrior



As the captain of a fleet of starships you seek to bring vital supplies to an orbiting space station. But every movement of your ships is fraught with danger, for the space station is menaced by constant asteroid showers. One by one you despatch your ships — will any of them survive?

The only problem in this program is caused by the asteroid movement. 260-290 create a random diagonal movement which is assigned to all the asteroids, so that they move as a body across the screen. The asteroids are moved in lines 14-18, and line 14 moves asteroids going off the top of the screen back to the

bottom. Line 16 moves asteroids going off the bottom back to the top. Line 17 is necessary because the left hand side screen locations are used by the Oric to define the foreground and background colours for that particular line. If an asteroid moved into these locations, the background and foreground colours for that line would be lost, and the entire line would turn black. Line 17 thus checks that the next memory location does not contain a character code lower than 32, as a number in this range must be the code for a colour. Any asteroids about to strike these locations are instead shifted across the screen to new positions.

Program notes

10 Keyboard click and cursor off.

14-18 Asteroid move routine, as explained above.

20 pokes the space station back into position in case an asteroid has just moved over it, and checks that the present starship position is not occupied by an asteroid.

30-90 Keyboard control routine.

150-230 Starship move routine. 150 omits this section if no key has been pressed. 160 checks the new position for asteroids or the space station. 170 dooms any ship with no fuel to drift helplessly while the asteroid shower continues. 180 ensures the ship does not move off-screen, 200 erases the ship from its old position and 210 draws it to its new one. 220 reduces the fuel and plots its amount on-screen, 230 completes the loop.

250-305 Game set-up routine. Used at the start of the game or whenever a starship is destroyed. 260 selects a random movement for the asteroid shower, 270-290 randomly position the asteroids on-screen and save their locations in the array A( ).

300-350 Game and routine. Various comments are offered and a score is given.

900-950 Initialisation routine.

960-1000 Offer instructions and clear screen for game start.

2000-2020 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

AD Asteroid direction of movement — to select value from D( ).

AM Asteroid movement — one of the 4 diagonal moves from D( ).

AN Asteroid's new position after movement.

AP Asteroid's present position.

AX Asteroid's x coordinate relative to BL at game-start.

AY Asteroid's y coordinate relative to BL at game-start.

FF Full fuel supply — starship fuel units at game-start.

MD My starship direction — to select value from D( ).

MP My starship position.

NA Number of asteroids.

NP My new starship position after movement.

SD Starships destroyed.

SF Starship fuel—units remaining.

SP Space station position.

ST Starship—number of starships that can be used.

Symbols used

AS Asteroid symbol.

ES Explosion symbol.

MS My symbol — a starship.

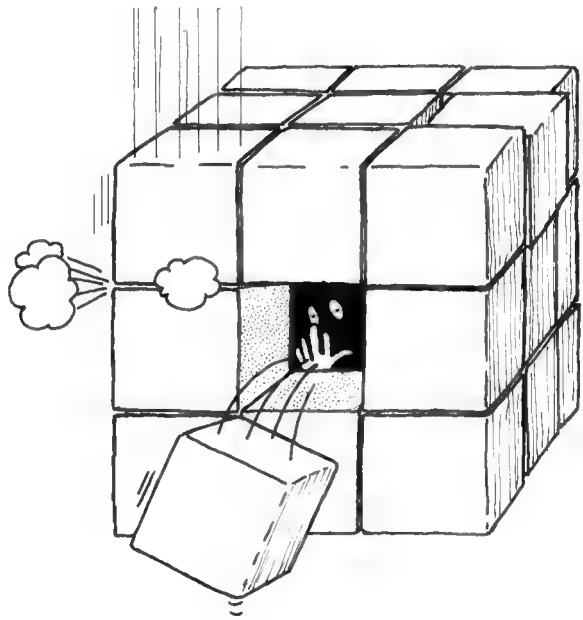
SS Space station symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
14 FORA=1TONA:AP=A(A):AN=AP+AM:IFAN<TLTHENAN=A
N+DL*LL
16 IFAN>BLTHENAN=AN-DL*LL
17 IFPEEK(AN)<32THENAN=AN+LL/2
18 POKEAP,32:A(A)=AN:POKEAN,AS:NEXT
```

```
20 POKESP,SS:P=PEEK(MP):IFP<>MSTHENPOKEMP,ES:G
OTO240
30 A$=KEY$:MD=8:IFA$="W"THENMD=0
35 IFA$="E"THENMD=1
40 IFA$="D"THENMD=2
50 IFA$="C"THENMD=3
60 IFA$="X"THENMD=4
70 IFA$="Z"THENMD=5
80 IFA$="A"THENMD=6
90 IFA$="Q"THENMD=7
150 IFMD=8THENNP=MP:GOTO170
160 NP=MP+D(MD):N=PEEK(NP):IFN<>32ANDN<>SSTHEN
POKEMP,ES:EXPLODE:GOTO240
170 IFSF=0THEN14
180 IFNP<TRORNP>BLTHEN220
200 POKEMP,32:IFN=SSTHENPOKENP,MS:GOTO320
210 MP=NP:POKEMP,MS
220 SF=SF-1:S$=STR$(SF):PLOT17,24,S$
230 GOTO14
240 SD=SD+1:IFSD=STTHEN310
250 PLOT5,24,"Spaceship destroyed":WAIT(500)
260 BD=2*INT(RND(1)*4)+1:IFBD=1THENBP=BL
270 CLS:FORA=1TONA:AX=INT(RND(1)*DT):AY=INT(RN
D(1)*DL)
280 IFBD=5THENBP=TR
290 AN=BL+AX-AY*LL:POKEAN,AS:A(A)=AN:NEXT
295 SP=TL+4*D(3):POKESP,SS:PLOT11,24,"Fuel:"
300 AD=2*INT(RND(1)*4)+1:AM=D(AD):SF=FF:MP=BR-
LL-1
305 POKEMP,MS:GOTO14
310 CLS:PRINT"You failed to reach the space st
ation.":GOTO350
320 CLS:PRINT"Well done! You reached the space
":PRINT"station with ship ";SD+1
330 PRINT"using ";FF-SF;" fuel units. This was
":PRINT"at skill level ";SL
340 PRINT"This is a total score of ";SL*(ST-SD
)*SF
350 END
900 DIMA(20):TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BE
ED:LL=40
910 AS=91:ES=92:MS=93:SS=94
915 FORZ=91TO94:GOSUB2000:NEXT
920 SD=0:ST=3:FF=300
```

```
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=(BL-TL)/LL
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK0:PAPER4:GOTO260
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA14,31,62,31,30,60,28,6,8,2,32,1,16,4,
0,0,12,12,12,12,12,63,45,45
2020 DATA0,12,18,63,63,18,12,0
5000 CLS:PRINT"You must try to land on the spa
ce":PRINT"station by guiding your ship"
5005 PRINT"through the asteroid showers. You "
:PRINT"begin at the bottom right, and"
5010 PRINT"the space station is near the top":
PRINT"left. You have ";ST;" ships each"
5020 PRINT"with ";FF;" units of fuel. The":PRI
NT"number of asteroids depends on "
5030 PRINT"skill level."
5035 PRINT:PRINT"Controls - use the keys aroun
d the":PRINT"letter S:"
5040 PRINTSPC(16);"Q W E":PRINTSPC(17);"A S D"
:PRINTSPC(18);"Z X C"
5050 PRINT:PRINT"Input your skill level, 1, ea
sy, to":PRINT"10,hard";
5060 INPUTSL:IFSL<1ORSL>10THEN5050
5070 NA=SL*2:RETURN
```

Trap



You control a little man who must constantly keep on the move. Any time he pauses too long in one place, blocks begin to appear around him. If he is too slow, he will be completely surrounded and unable to move.

This is definitely the game for those who fancy they have quick reflexes! One part of the game which might tend to slow things down is the necessary routine that checks if the man is completely surrounded, in which case we want the game to end. It might seem that we must look at all 8 positions surrounding the man at every 'turn', but this is not the most efficient method. It will be quicker if the positions around the man are only examined until a space is found. If there is a space nearby, the man can still escape, and the game continues. Lines 160-190 carry out this procedure,

and by only looking at the surrounding positions until a space is found, the game is speeded up.

Program notes

10 Keyboard click and cursor off.

20-100 Keyboard control routine.

110-115 Player move routine. 110 checks if the new position is a wall or block, and 115 pokes the man to the new position if it is clear.

125-140 Block position routine. 125 omits this section if the skill level is too low, 130 chooses a random position for the block, and 140 pokes it next to the man if the skill level is high enough.

160-200 Man surrounded routine. Each position around the man is checked until a space is found. If no space is found 200 will terminate the game.

220-230 Time and pause whose length depends on the skill level.

250-280 Game end routine, including the score.

900-950 Initialisation routine.

960-1110 Game set-up routine. 1100-1105 draw the walls around the playing area, 1110 pokes the man on-screen.

2000-2010 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

CM Computer movement — used to pick movement from D( ).

CV Computer velocity — high values give slow games.

MD Man direction — used to pick movement from D( ).

MP Man position.

NC New block position.

NP Man's new position.

Symbols used

BS Block symbol.

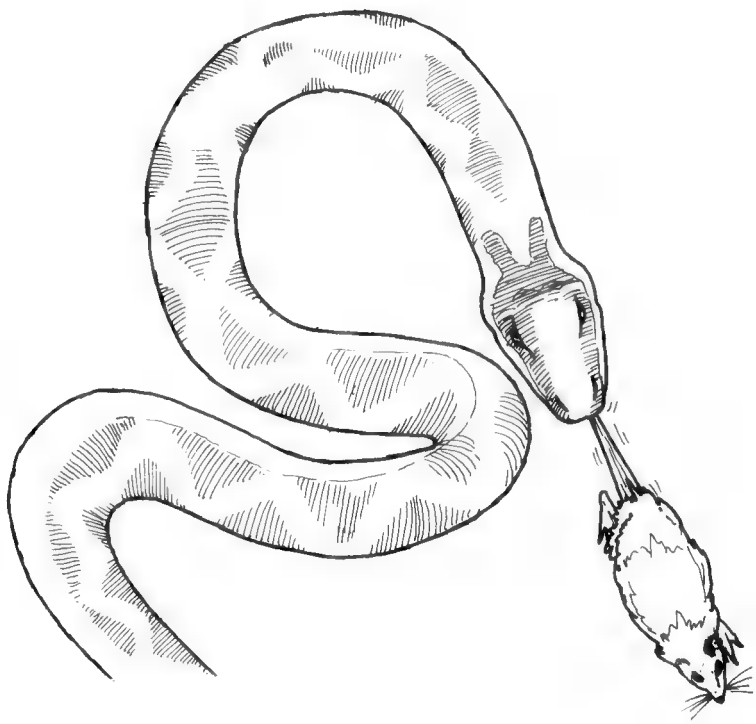
MS Man symbol.

WS Wall symbol.

```
10 PRINTCHR$(17)CHR$(6):GOTO900
20 MD=8:A$=KEY$:IFA$="W"THENMD=0
30 IFA$="E"THENMD=1
40 IFA$="D"THENMD=2
50 IFA$="C"THENMD=3
60 IFA$="X"THENMD=4
70 IFA$="Z"THENMD=5
80 IFA$="A"THENMD=6
90 IFA$="Q"THENMD=7
100 IFMD>7THEN130
110 NP=MP+D(MD):P=PEEK(NP):IFP=WSORP=BSTHEN1
30
115 POKEMP,OS:MP=NP:OS=P:POKEMP,MS
125 IFRND(1)>SLTHEN220
130 CM=INT(RND(1)*8):NC=MP+D(CM)
140 IFRND(1)<SLTHENPOKENC,BS
160 A=0
170 CM=D(A):NC=MP+CM:CN=PEEK(NC)
180 IFCN=32THEN220
190 IFA<7THENA=A+1:GOTO170
200 GOTO250
220 T=T+1:PLOT18,24,STR$(T)
230 WAIT(CV):GOTO20
250 CLS:PRINT"Your time to blockade was ";T
270 PRINT"At skill level ";SL*10
275 PRINT"A total score of ";SL*10*T
280 END
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEEC:LL=4
0
910 MS=91:BS=92:WS=93
915 FORZ=91TO93:GOSUB2000:NEXT
920 T=0:NB=0:TF=20
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:CP=TL+INT(DT/2)+6*LL:M
P=BR-LL-1
```

```
960 CLS:PRINT"Do yõu want instructions (Y/N)
";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:
NEXT
1110 PLOT12,24,"TIME:":POKEMP,MS:GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,
U:NEXT:RETURN
2010 DATA14,14,4,31,4,4,10,17,63,33,33,33,33
,33,33,63,63,63,63,63,63,63,63
5000 CLS:PRINT"In Trap you must try to escap
e":PRINT"walls which will surround you"
5010 PRINT"If you linger too long in one":PR
INT"spot. Try to keep free for as long"
5020 PRINT"as possible. When you are":PRINT"
completely surrounded the game ends."
5050 PRINT:PRINT"Controls - use the keys aro
und the":PRINT"5 key to move:"
5060 PRINTSPC(16);"Q W E":PRINTSPC(17);"A S
D":PRINTSPC(18);"Z X C"
5070 PRINT:PRINT"Input your skill level, 1,
easy":PRINT"to 10, hard";:INPUTSL
5080 IFSL<1ORSL>10THEN5050
5090 CV=11-SL:SL=SL/10:RETURN
```

Ratkiller



In this game you are a snake and you must try to kill a rat by running into it and biting it with your venomous jaws. However, the snake is continually moving, and you will perish if you bite yourself or touch the electrified walls surrounding you . . .

In this game the parts of the snake's body are stored in an array, the tail being stored as S(1) and so on. The highest array value thus contains the present position of the snake's head. Whenever the snake moves the tail position is erased and a new segment is added on at the snake's head, thus keeping the snake a constant

length. As in the earlier program, Caterpillar, the snake is moved through an array to keep a constant record of positions of all parts of the body. Note that if you choose a long snake to make hunting the rat easier, you will find it moves very fast and you will often miss your target.

Program notes

10 Keyboard click and cursor off.

20-80 Keyboard control routine. Horizontal and vertical movements only are catered for.

110-160 Snake move routine. 110 finds the new snake position and checks if you have bitten yourself. 120 checks if you have got the rat, and 130 checks for hitting the electrified wall. 135-145 move the snake to its new position, 150 updates the time, and 160 decides on the basis of skill level whether or not the rat should move this time.

170-190 Rat move routine. 170 selects a random movement from the array D( ), and checks that the new position is not a wall, or the snake, 180. 190 pokes the rat to its new position.

200-280 Game end routine. Various messages are given depending on the outcome of the game, as well as the score.

960-1210 Game set-up routine. 1100-1105 pokes the electrified walls to the screen, 1110-1120 choose a random position for the rat and poke it to the screen. 1170 picks a random position for the snake's head, and 1175 checks that this is empty. 1180 pokes the head and the rest of the snake's body on-screen, continuing in that direction until a position is occupied and a new direction for the body must be found, 1195. 1210 begins the game.

2000-2030 User defined character creation routine.

5000 onwards — instructions, selection of snake length and setting of skill level.

Important variables

BS Biggest snake length allowed.

LS Selected snake length.

MD My direction of movement — to select a value from array D( ).

NP New head position for snake.

RM Rat movement — to select a value from array D( ).

RN New rat position.

RP Present rat position.

RX Rat's x coordinate relative to BL.

RY Rat's y coordinate relative to BL.

ST Snake tail position.

SX Snake's x co-ordinate relative to BL.

SY Snake's y co-ordinate relative to BL.

TS Tiniest snake length — smallest selectable snake length.

Symbols used

H Snake head symbol, varying with movement direction.

RS Rat symbol.

SS Snake symbol.

WS Wall symbol.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 A$=KEY$:IFA$="W"THENMD=0:H=93
40 IFA$="D"THENMD=2:H=94
60 IFA$="X"THENMD=4:H=95
80 IFA$="A"THENMD=6:H=96
110 NP=SH+D(MD):P=PEEK(NP):IFP=SSTHEN210
120 IFP=RSTHEN230
130 IFP=WSTHENWH=1:GOTO270
135 S=S+1:ST=ST+1:IFST>900THENST=0
140 IFS>900THENS=0
145 S(S)=NP:POKENP,H:POKENP-D(MD),SS:POKES(ST)
,32
150 T=T+1:PLOT17,24,STR$(T)
160 SH=NP:IFRND(1)>SLTHEN20
```

```
170 RM=INT(RND(1)*8):RN=RP+D(RM):R=PEEK(RN):IF
R=WSTHEN20
180 IFR=SSTHEN230
190 POKERP,32:RP=RN:POKERP,RS:IFT<TETHEN20
200 PLAY0,0,0,0:WAIT(300):CLS:PRINT"Your time
has run out.":GOTO270
210 PLAY0,0,0,0:WAIT(300)
220 CLS:PRINT"In your excitement you bit yours
elf":PRINT"and died.":GOTO270
230 PLAY0,3,1,1000:WAIT(300):CLS:PRINT"You sei
ze the rat in your poison"
235 PRINT"fangs and inject it with poiso^"
240 PRINT"in time ";T;" at skill level ";SL*10
265 PRINT"A total score of ";(TE-T)*SL*10
270 IFWH=1THENCLS:PRINT"You blundered into a w
all and killed":PRINT"yourself."
280 END
900 DIMS(900):TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£B
EED:LL=40
910 RS=91:SS=92:H=95:WS=123
915 FORZ=91TO96:GOSUB2000:NEXT:Z=WS:GOSUB2000
920 T=0:TE=400:MD=4:RD=0:TS=4:BS=19:ST=0
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:SZ=BL+2*D(1)
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS:INK0:PAPER2
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
1110 TD=DT-4:SD=(DL-4*LL)/LL:RX=INT(RND(1)*TD+
1)
1120 RY=INT(RND(1)*SD+1):RP=SZ+RX-RY*LL:POKERP
,RS
1170 SX=INT(RND(1)*TD+1):SY=INT(RND(1)*SD+1):S
H=SZ+SX-SY*LL
1175 IFPEEK(SH)<>32THEN1170
1180 S(LS)=SH:SP=SH:POKESP,SS:FORA=LS-1TO1STEP
-1
1190 NP=SP+D(RD)
1195 IFPEEK(NP)<>32THENRD=INT(RND(1)*8+1):GOTO
1190
```

```
1200 SP=NP:S(A)=NP:POKESP,SS:NEXT
1210 SH=S(LS):PLAY0,1,4,100:GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA18,30,12,63,12,12,30,51,12,12,18,51,5
1,18,12,12,0,0,0,0,33,18,12,12
2020 DATA0,4,8,48,48,8,4,0,12,12,18,33,0,0,0,0
,0,8,4,3,3,4,8,10
2030 DATA63,63,63,63,63,63,63,63
5000 CLS:PRINT"You are a hungry snake chasing"
:PRINT"a rat. Your poison is so deadly"
5005 PRINT"that even to touch your skin is":PR
INT"fatal - so don't bite yourself!"
5010 PRINT"You must catch and kill the snake":
PRINT"in time ";TE
5015 PRINT"Long snakes can encircle the rat":P
RINT"more easily but you are more"
5020 PRINT"likely to bite yourself."
5030 PRINT:PRINT"Avoid hitting the walls - at
the":PRINT"speed you're moving hitting'
5035 PRINT"your head will probably be fatal!"
5040 PRINT:PRINT"Controls - use the keys aroun
d the S:":PRINTSPC(18);"W"
5045 PRINTSPC(16);"A   D":PRINTSPC(18);"X"
5050 PRINT:PRINT"How long do you want to be, f
rom":PRINTTS;" units to ";BS;
5060 PRINT"units";:INPUTLS
5070 IFLS<TSORLS>BSTHEN5050
5080 PRINT:PRINT"Input skill level, 1 for slow
 rats":PRINT"up to 10 for very fast";
5090 INPUTSL:IFSL<1ORSL>10THEN5080
5100 S=LS:SL=SL/10:RETURN
```

Catch

In Catch numbers bounce around the screen and off obstacles. If you can catch the numbers they are added to your score. At the higher skill levels there are an awful lot of obstacles and things can get very difficult!

Programming the movement of the number as it bounces off obstacles is fairly straightforward, because to make the number harder to catch its movement should be unpredictable. A number must sometimes bounce in a different direction if it hits the same obstacle more than once, otherwise a player need only 'lie in wait' for the number to catch it. Line 160 thus chooses a completely

random movement for the number ball whenever it strikes an obstacle.

For similar reasons the number's initial appearance on-screen must be unpredictable, and lines 260-290 ensure that it appears randomly from one of the four corners of the playing area.

Program notes

10 Keyboard click and cursor off.

20-100 Keyboard control routine. The player tries to catch the number by moving in any of the 8 major compass directions.

110-130 Player move routine. 110 finds the new position and checks that it is not a wall. If it is not, it must be either a space or a number, and 120 adds this number to the score if necessary. 130 pokes the player's symbol to its new position.

150-210 Number move routine. 150 finds the new position and erases the number from its previous position. 160 picks a random direction of movement when an obstacle is hit. If the number hits the player's symbol, the number value is added to the score, 170. 180 moves the ball to its new position, 190 shows the time taken so far, 200 is a pause whose length depends on the skill level, and 210 ensures that the whole procedure is repeated until the time limit is exceeded.

220-230 Game end routine, giving the score.

250-320 Ball serve routine. 250 checks that not all the balls have been used, 260 chooses a random diagonal movement for the number, and the remainder of 260 and lines 270-290 pick the appropriate corner from which the number is 'served'. 300 finds the number's new position and 320 places it there.

900-950 Initialisation routine.

960-1160 Game set-up routine. 1100-1105 draws the sides of the playing area, 1120-1150 pokes random obstacles on-screen, their number depending on the skill level. 1160 pokes the player's symbol on-screen.

2000-2010 User defined character creation routine.

5000 onwards — instructions and setting of skill level.

Important variables

BD Ball direction of movement — used to choose value from D( ).

BV Ball velocity. High values slow ball.

MD My direction of movement — used to choose value from D( ).

MP My position.

NB Number ball new position.

NP My new position.

NW Number of obstacle walls.

S Score.

WP Wall position for obstacles.

WX Wall x coordinate relative to BL.

WY Wall y coordinate relative to BL.

Symbols used

BS Number ball symbol — set to character code for numbers 0-9.

MS My symbol.

OS Zero symbol — character code for number 0.

TS Top symbol — character code for number 9.

WS Wall symbol — used for both walls and obstacles.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 REPEAT:A$=KEY$:MD=8:IFA$="W"THENMD=0
30 IFA$="E"THENMD=1
40 IFA$="D"THENMD=2
50 IFA$="C"THENMD=3
60 IFA$="X"THENMD=4
70 IFA$="Z"THENMD=5
80 IFA$="A"THENMD=6
90 IFA$="Q"THENMD=7
```

```
100 IFMD>7THEN150
110 NP=MP+D(MD):P=PEEK(NP):IFP=WSTHEN150
120 IFP<>32THENS=S+SL*(P-OS):POKENP,32:GOTO250
130 POKEMP,32:MP=NP:POKEMP,MS
150 NB=BP+D(BD):B=PEEK(NB):POKEBP,32
160 IFB=WSTHENBD=2*INT(RND(1)*4)+1:GOTO150
170 IFB=MSTHENS=S+SL*(BS-OS):GOTO250
180 BP=NB:POKEBP,BS
190 T=T+1:PLOT17,24,STR$(T)
200 WAIT(BV)
210 UNTILT>=TE
220 CLS:PRINT"You scored ";S;" points at skill
 level ";SL
230 END
250 PING:IFBS<TSTHENBS=BS+1
260 BD=2*INT(RND(1)*4)+1:IFBD=1THENBP=BL
270 IFBD=3THENBP=TL
280 IFBD=5THENBP=TR
290 IFBD=7THENBP=BR
300 BP=BP+D(BD):IFPEEK(BP)<>32THEN260
320 POKEBP,BS:GOTO20
900 TL=£BBD2:TR=£BBF5:BL=£BECA:BR=£BEED:LL=40
910 OS=48:TS=57:BS=OS:MS=91:WS=92
915 FORZ=91TO92:GOSUB2000:NEXT
920 S=0:T=0:TE=200
930 D(0)=-LL:D(1)=-LL+1:D(2)=1:D(3)=LL+1
940 D(4)=LL:D(5)=LL-1:D(6)=-1:D(7)=-LL-1
950 DT=TR-TL:DL=BL-TL:MP=BL-LL+1
960 CLS:PRINT"Do you want instructions (Y/N)";
965 INPUTA$:IFA$="Y"THENGOSUB5000:GOTO1000
980 GOSUB5050
1000 CLS
1100 FORA=TLTOTR:POKEA,WS:POKEA+DL,WS:NEXT
1105 FORA=TLTOBLSTEPLL:POKEA,WS:POKEA+DT,WS:NE
XT
1120 DT=DT-6:DL=(DL/LL)-6:FORA=1TONW
1130 WX=INT(RND(1)*DT+3):WY=INT(RND(1)*DL+3):W
P=BL+WX-WY*LL
1140 IFPEEK(WP)<>32THEN1130
1150 POKEWP,WS:NEXT
1155 PLOT11,24,"TIME:"
1160 POKEMP,MS:GOTO250
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
```

```
2010 DATA63,51,45,30,30,45,51,63,63,63,63,63,6
3,63,63,63
5000 CLS:PRINT"You must try to catch the numbe
rs":PRINT"bouncing on the screen."
5005 PRINT"The first number is 1: after that":
PRINT"numbers increase by 1 until 9 is"
5010 PRINT"reached. To make things harder, the
":PRINT"speed of the ball and the"
5020 PRINT"number of obstacles on screen also"
:PRINT"vary with the skill level."
5030 PRINT:PRINT"Controls - use the keys aroun
d the":PRINT"S to move:'
5040 PRINTSPC(16);"Q W E":PRINTSPC(17);"A S D"
:PRINTSPC(18);"Z X C"
5050 PRINT:PRINT"Input your skill level, 1, ea
sy, to":PRINT"10,hard";
5060 INPUTSL:IFSL<1ORSL>10THEN5050
5070 NW=SL*2:BV=11-SL:RETURN
```

Canyon Bomber

In this game you must drop bombs into a canyon filled with numbers. Each number hit is added to your score. Bombs fall at an angle, and the game becomes quite difficult when there are just a few numbers left, as you are only allowed a limited number of bombing runs which produce no addition to your score.

The setting up of the canyon is a complex matter, as it is randomly drawn each time, and the screen locations for the top and bottom of the canyon must be determined. The canyon must also not contain any 'blind' spots which cannot be bombed.

As the canyon is constructed in subroutine 3000, a column of numbers is drawn over each position. As the bombs fall at an angle, it is easy to knock the middle out of a column of numbers. To leave the remaining numbers hanging in the air would be unattractive, to say the least!

So every time a bomb strikes, line 140 moves all the numbers above the strike position down one line. This involves a lot of peeking and poking, so in this program a running score is omitted to help speed things up. A final score is, however, given at the end.

Program notes

10 Keyboard click and cursor off.

20-50 Plane move routine. 20 finds the plane's new position and checks if it has reached the right or left of the screen. If so, 25 reverses the flying direction, and the appropriate plane symbol is picked, the remainder of 25, and 26. 30 checks to see if the previous run across the screen produced a scoring hit. 40 adds one to the bomb misses if necessary, and checks if the allowable miss total has been reached. 50 then pokes the plane to its new position.

60-110 Bomb drop routine. BF is the bomb flag, if in 60 this is 1, a bomb is dropping and another may not be dropped until this one explodes. 70 is a keyboard control to see if you wish to drop a bomb. 80 sets the initial bomb position to that of the plane, and 90 moves the bomb to a new position and checks if the canyon wall has been hit. 100 draws the bomb at its new position and then erases it. If the bomb has moved to an empty position, the program loops back to move the plane again, 110.

120-150 Rearrangement of numbers plus scoring routine. 120 adds the number hit to the score, and notes that the number hit is now one more. 140 moves down all the numbers above the one hit so that the empty space is filled, 150 continuing the movement until all the numbers in that particular column have been moved down.

155-170 Score: this is doubled if all the numbers have been hit, 155.

900-1005 Initialisation of variables. This section is longer than usual because it involves the random determination of the positions of the left and right cliff-top sections of the canyon, plus the left and right canyon bottom positions. 1000 makes sure the canyon bottom is not off-screen.

1010-1100 Game set-up routine. 1020-1100 draw the canyon, all using sub-routine 3000 to draw different parts of the canyon. 1080 gets rid of two excess numbers introduced in the process, 1100 pokes the plane to its start position and play begins.

2000-2020 User defined character creation routine.

3000-3050 Draw canyon subroutine. This also puts a number stack above each part of the canyon. First the canyon floor is poked into position, 3000. 3005 and 3010 between them randomly determine the first number to be poked onto this part of the canyon. Successive numbers poked above this will be of lower value. 3030 counts the numbers as they are poked onto the screen, and calculates the position and value for the number to be poked above the present one. If this value is one or more, and the position is not above the right-hand cliff top, this number can also be poked to the screen, 3040. Otherwise the program will move to the next position along the canyon and repeat the process, 3050.

5000 on — instructions.

Important variables

BD Bomb direction of movement — diagonally left or right.

BF Bomb flag — set to 1 when bomb is dropped.

BH Bomb hit flag — set to 1 when hit is made.

BM Bomb miss count.

BP Bomb present position.

BT Total bomb misses allowed.

CB Biggest depth canyon can have. This cannot exceed half the distance between the two cliff tops, otherwise it becomes difficult to join the bottom of the canyon to the other cliff top.

CD Canyon depth.

CL Canyon base left position.

CM Minimum canyon depth.

CR Canyon base right position.

DC Distance vertically of canyon top from TL.

DT Distance between top left and top right of screen.

LB Biggest length cliff top on right can have.

LM Minimum length cliff top on left can have.

LE Left end position of cliff top.

LS Left start position of cliff top.

ML Canyon left side slope, as well as bomb drop movement when plane moves left to right.

MR Canyon right side slope, as well as bomb drop movement when plane moves right to left.

NB New bomb position.

NC Count for numbers poked at start.

NH Count for how many numbers have been hit.

NP New plane position.

PM Plane movement.

PN Position one above NB.

PP Plane position.

S Score.

RB Biggest length cliff top on right can have.

RM Minimum length cliff top on right can have.

RE Right end position of cliff top.

RS Right start position of cliff top.

Symbols used

BS Bomb symbol.

OS Nought symbol — character code for number zero.

NS Nine symbol — character code for number nine.

PS Plane symbol — either to right, P1, or left, P2.

WS Wall symbol — used for cliff top and canyon sides.

```
10 PRINTCHR$(6)CHR$(17):GOTO900
20 NP=PP+PM:IFNP<>TLANDNP<>TRTHEN50
25 PM=-PM:BD=2*LL-BD:IFPM<0THENPS=P2
26 IFPM>0THENPS=P1
30 IFBH=1THENBH=0:GOTO50
40 BM=BM+1:IFBM=BTTHENCLS:GOTO160
50 POKEPP,32:PP=NP:POKEPP,PS
60 IFBF=1THEN90
70 A$=KEY$:IFA$<>"B"THEN20
80 BP=PP:BF=1:DB=BD
90 NB=BP+DB:PB=PEEK(NB):IFPB=WSORPEEK(NB-LL)=W
STHENBF=0:EXPLODE:GOTO20
100 BP=NB:POKEBP,BS:WAIT(1):POKEBP,32
110 IFPB=32THEN20
120 S=S+PB-OS:BH=1:NH=NH+1:IFNH=NCTHENS=S*2:GO
TO155
140 PN=NB-LL:PB=PEEK(PN):POKENB,PB:IFPB=32THEN
20
150 NB=PN:GOTO140
155 CLS;PRINT"You hit them all and double your
":PRINT"score."
160 PRINT"You scored ";S;"points."
170 END
900 TL=£BC9A:TR=£BCBD:BL=£BECA:BR=£BEED:LL=40
910 OS=48:NS=57:P1=91:BS=92:WS=93:P2=94:PS=P1
915 FORZ=91TO94:GOSUB2000:NEXT
920 S=0:BM=0:NH=0:BT=3:LB=5:LM=1:RB=5:RM=1:DC=
1:CM=7
945 PP=TL:PM=1:BD=LL+1:BF=0:BH=0
950 DT=TR-TL:LS=TL+LL*DC:LE=LS+INT((LB-LM)*RND
(1)+LM)
960 RE=LS+DT:RS=RE-INT((RB-RM)*RND(1)+RM)
970 ML=LL+1:MR=LL-1:L=TL+DC*ML:R=TR+DC*MR:IFLE
<LTTHENLE=L
```

```
980 IFRS>RTHENRS=R
990 CB=INT((RS-LE)/2)
1000 CD=INT((CB-CM)*RND(1)+CM):IFLE+CD*LL>BLTH
EN1000
1005 CL=LE+CD*ML:CR=RS+CD*MR
1010 CLS:PRINT"Do you want instructions (Y/N)"
;
1015 INPUTA$:IFA$="Y"THENGOSUB5000
1020 INK0:PAPER4:CLS:B=LS:C=LE:ST=1:GOSUB3000
1040 B=LE:C=CL:ST=ML:GOSUB3000
1050 B=CL+1:C=CR-1:ST=1:GOSUB3000
1060 B=RS:C=CR:ST=MR:GOSUB3000
1070 B=RS:C=RE:ST=1:GOSUB3000
1080 POKELE,32:POKERS,32:NC=NC-2
1100 BD=ML:POKEPP,PS:GOTO20
2000 XX=46080+Z*8:FORO=XXTOXX+7:READU:POKEO,U:
NEXT:RETURN
2010 DATA16,8,36,63,4,8,16,0,0,10,4,4,4,4,0,0,
63,63,63,63,63,63,63,63
2020 DATA2,4,9,63,8,4,2,0
3000 FORA=BTOCSTEPST:TS=NS:P=A:POKEA+LL,WS
3005 IFC=LEORC=REORPEEK(P)<>32THEN3050
3010 IFRND(1)<.5THENTS=TS-1
3020 IFRND(1)<.5ANDTS>OS+1THEN3010
3030 NC=NC+1:POKEP,TS:IFTS>OS+1THENP=P-LL:TS=T
S-1
3040 IFTS>OS+1ANDP>RSTHEN3030
3050 NEXT:RETURN
5000 CLS:PRINT"You control a bomber flying ove
r":PRINT"a canyon. On each pass you"
5010 PRINT"must hit at least one number in":PR
INT"the canyon, whose value will"
5020 PRINT"added to your score. You may miss";
PRINT"just ";BT;" times, then the game"
5030 PRINT"ends. You may only drop one bomb at
":PRINT"a time, and no others can be"
5040 PRINT"dropped while that one is in flight
":PRINT"The bomb will fall at an angle'
5050 PRINT"Controls - press B to drop a bomb."
:PRINT"Press any key to begin."
5060 REPEAT:GETA$:UNTILA$<>" ":RETURN
```

Other titles available from Micro Press:

THE MICRO MAZE: A GUIDE TO PERSONAL COMPUTING

Wynford James

0 7447 0000 0

QUALITY PROGRAMS FOR THE BBC MICRO

Simon

0 7447 0001 9

15 GRAPHIC GAMES FOR THE SPECTRUM

Richard G. Hurley

0 7447 0002 7

SIMON'S PROGRAMS FOR THE ELECTRON

Simon

0 7447 0004 3

20 GAMES FOR THE ORIC-1

These 20 diverse and highly entertaining games are self contained and easy to enter, and will provide hours of pleasure, using the capabilities of your ORIC to the full. The process of entering the programs will provide insight into the many and varied techniques used in the compilation of programs of this sort. By exploring fully the challenges of computer programming with this book, the reader not only finds games of lasting interest, but may gain insight into sound programming techniques, to be able to originate and write his own games in the future.

The Author

Wynford James was until recently a maths teacher, actively involved in the development of computer studies throughout his school. He uses his experience as a teacher in communicating clearly and in an interesting manner in the writing of this book. Wynford is now a technical author with ICL.

GB £ NET +005·95

ISBN 0-7447-0003-5

00595

9 780744 700039

20 GAMES FOR THE ORIC-1 Wynford James